China-pub.com
下载

第7章 高 级 报 表

既然已经完全懂得了表单，现在就可以深入研究下一个大的题目：报表。可以用 Oracle Developer Reports创建非常简单或者非常复杂的报表。由于报表中不同种类对象的数量要比表单中少，所以可以用很多不同的方式对它们进行组合。另外， Oracle Developer提供的服务器工具可用于在因特网上发布报表，这样可以很容易地跨公司或跨世界地发布报表。第 4章介绍了用Web Wizard建立用于在因特网上发布的 HTML及Adobe Acrobat报表。本章详细介绍Oracle Developer Reports的主要数据模型、布局及报表特点。本章是在第 4章的指导性介绍及第5章的报表处理综述的基础上再进行的详细介绍，可帮助建立几种可用的基本报表类型。

注意 Oracle Developer Reports是一个很方便的工具。本章概要介绍该工具，并介绍一些使用广泛的报表类别的基本报表策略，但可以使用Report Builder构造大量的各种不同的报表。利用这里学习的工具进行实验，考虑如何设计所需要的所有不同类型的报表。如果愿意，可以购买独立于Oracle Developer的Reports产品，这样即使不需要建立表单，也可以利用Oracle Developer技术比较经济地建立报表。Oracle Reports还集成了Oracle Express，它是一种数据探测工具，是对于Express的数据挖掘特征的强有力的扩充。

7.1 报表数据模型

在第4章已经看到了如何建立具有单个查询并加入了汇总分组的基本数据模型。本节详细介绍在数据模型中如何建立和使用分组、公式与汇总列、多重查询以及数据链接。后面各节的特殊类型报表都基于此。要以图形形式查看数据模型，单击 Data Model Editor工具或选择Tools|Data Model Editor 菜单项。

7.1.1 分组

每个Oracle Developer Reports是由一些顺序排列的或结构化的列组成的。查询把这些列，即数据库列(database columns)与数据库中的数据联系起来。分组 (group)是一个组织报表各列的数据模型对象，报表列包括数据库列、公式列、汇总列及占位列。

例如，第4章的Ledger报表查询Talbot数据库中的Ledger表。在这个例子中，该查询采集Ledger表中的数据库列。在报表中只有一个分组，是由 Ledger表中的所有行组成的。提供总计数量的汇总数据是在分组级的一个汇总列。

有两种类型的分组 —— 记录组及分隔组。记录组 (record group)对应查询返回的所有记录。在很大程度上它类似于表单的基本表数据块，记录组提供了构成报表基本数据的列簇。分隔组是一个位于记录组和它的查询之间的分组。分隔组为报表提供了一种层次结构，这通常是为了把记录组中的记录分隔成分别进行汇总的段。详细内容参看 7.3.1节的“汇总列”。每个分隔组至少有一列具有Break Order属性(及相对应的Set Break Order 属性)设置。

图7-1给出了第4章Ledger Summary报表的数据模型。 LedgerGroup是一个由数据库的

China-pub.com
下载

ActionDate列、Item列、Quantity列、Rate列及Amount列组成的记录组，所有这些列都来自于Ledger表。该记录组包含报表的基本数据，数据都是通过LedgerQuery从数据库获得的。PersonGroup作为一个分隔组，为报表增加了一层，它位于LedgerGroup及LedgerQuery之间。数据库的Person列具有Break Order属性设置，是分隔组的分隔列。Subtotal列是一个汇总列，它对报表中的每个分隔组分别提供一个部分和(参看本章后面的“汇总列”)。

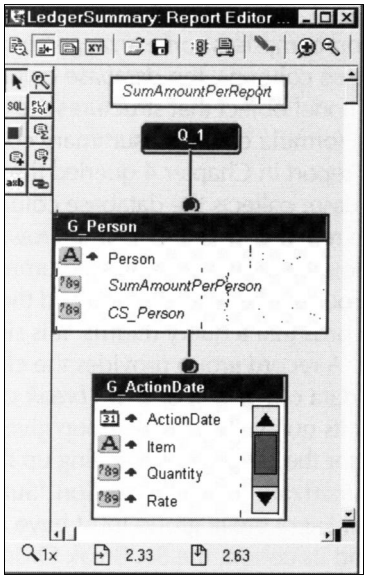


图7-1 Ledger Summary数据模型

如果双击“分组”，在Data Model Editor或在Object Navigator中可以看到分组Property Palette中有限的属性。除了分组的名字和注释属性之外，只有一个其他属性，即Filter Type(过滤器类型)。可以把这个下拉列表设成None、First、Last或PL/SQL。

如果把该属性设成First或Last，将会看到另一个属性Number of Records魔术般地出现了。如果在属性中输入一个数值，Oracle Developer会过滤掉此数值对应的那个记录后面或前面的所有记录。例如，如果设置Number of Records为10并且Filter Type为First，不管返回多少条查询，则只能看到前10条记录。

如果设置该属性为PL/SQL，另一个属性出现了：PL/SQL Filter。单击此属性会显示Program Unit Editor，并带有一个返回布尔值(True或False)的函数模板。可以用自己愿意的任何逻辑来决定是返回True还是False。

当Oracle Developer每取一行数据时，它会调用该函数。如果函数返回False, Developer将从报表中丢弃此记录，如果返回True，则会在报表中包含该记录。关于函数及函数在过滤器和公式列中应用的详细内容，参看第12章“函数”部分。

7.1.2 公式列

第4章的Ledger Summary报表建立的数据模型有数量列、价格列及总计列。代替从数据库中取后一列，用数量乘以价格来计算。在报表中，公式列(formula column)的值是从函数获得而不是直接从数据库获得。本节介绍如何建立一个公式列来计算总计数据。

用PL/SQL定义公式列的值，可以完成各种功能。大多数功能可以用SQL或数据库PL/SQL函数实现。当要实现一些不易用SQL实现的事情时，通常是用PL/SQL，例如需要进行条件判断的计算(IF-THEN- ELSE)。但如果不是这种情况，为什么还要用报表函数实现呢？有三方面原因。

第一，在应用程序中通常要计算一些只对本应用程序有意义的值。如果该值对于几个不同的应用程序均有用，可能希望把这个计算移到服务器上来共享，避免一遍又一遍地编写代码。如果用Oracle SQL可以计算该值，当然不需要用函数实现它。例如，用如下SQL可以避免使用公式列：

```
SELECT Person, ActionDate, Item,
       Quantity, Rate, Quantity * Rate "Amount"
FROM Ledger
```

China-pub.com 下载

第二，如果希望报表独立于Oracle，则不能使用Oracle提供的大多数高级特性，如字符串及日期函数。如果将计算放进报表的 PL/SQL函数中，即使非常复杂的报表，通过 ODBC也可以访问其他非Oracle数据库。

第三，把计算移到服务器上，在一定程度下会提高客户端的性能。如果这是把 PL/SQL放进数据库的原因，就一定要完全测定该应用程序的基准点。

下面一段教程指导如何在Data Model Editor中建立Amount公式列。

1) 双击LedgerQuery并像下面这样修改它：

```
SELECT Person, ActionDate, Item, Quantity, Rate
  FROM Ledger
```

修改查询，删除数据库的 Amount列以及当前基于它的无效的任何汇总域。

2) 双击Data Model Editor中的Formula Column工具，然后单击 Rate列下的分组框。

3) 双击产生的resulting column CF_1查看Formula Column Property Palette(见图7-2)。

注意 在Data Model Editor的旁边,如果看不到带有Formula Column工具的工具板，则将焦点放在Data Model Editor上，选择View/Tool Palette菜单项。

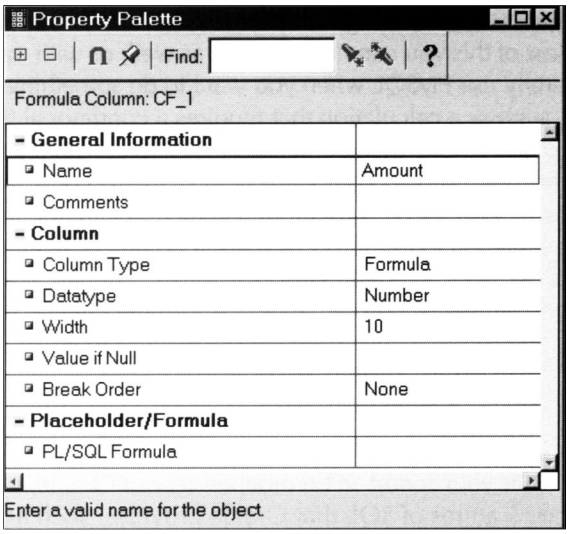


图7-2 公式列属性板

4) 在General分组中把名字改成Amount。

5) 单击PL/SQL Formula属性，显示计算该值的PL/SQL函数的Program Unit Editor。

6) 增加一个语句来计算总和(见图7-3)：

```
return :Quantity * :Rate;
```

冒号表明是引用报表项目。

7) 单击Compile并处理错误，然后单击Close。

当单击属性表中的Ok时，在LedgerGroup中会看到新列 Amount。在Editor中该列以斜体形式显示，以便知道它是一个公式列(见图7-4)。

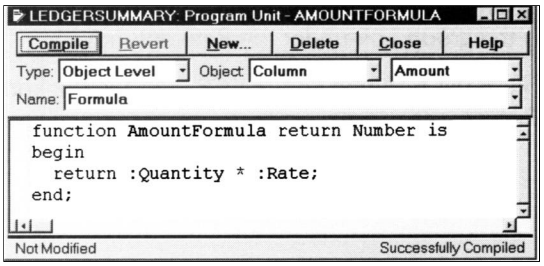


图7-3 增加一个语句来计算总和

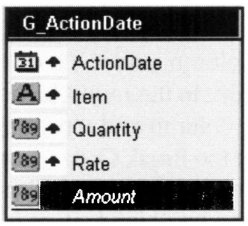


图7-4 LedgerGroup中的新列Amount

切换到Object Navigator中，在Data Model部分的Groups下的LEDGERGROUP下面会发现

China-pub.com
下载

一个新的列。公式列有两个图标，一个表示公式列，一个表示PL/SQL代码。单击Formula Column图标会显示该列的Property Palette，或者单击PL/SQL图标显示公式列的PL/SQL代码(见图7-5)。

图7-5 新的列

注意 当分组中的所有数据是空值并且想压缩标准打印时，实际上把公式列当成了一个占位符。

关于函数及在过滤器和公式列中使用的详细内容参见第 12章“函数”。

正如最后一步，需要重新建立增加的 Amount列和汇总列之间的关系，这关系是因为列的改变而变得无效的。下一节讲解如何设置汇总列的 Source属性，以及如何在报表中增加一个新的汇总列。

提示 要使格式正确，通常最好的办法是返回到Report Wizard，按照希望的形式重新建立所有的列，然后在Live Previewer中重新格式新的Amount列。

7.1.3 汇总列

在报表的任何地方，公式列都允许计算并显示基于一条记录的值。然而，很多时候，需要在一组记录的基础上完成一些复杂的计算。汇总列 (summary column)便是用于计算一个聚集值的数据对象。

分隔列(break column)是从记录组移到较高级分隔组的列。该列告诉 Developer：对于该列的每个不同值创建一个分隔或组进程。也就是，当报表取一条记录时，它会检查分隔列的值，如果与前一条记录不同，就会分开并完成一些特殊动作。在报表布局中，这对应着重复框架对象。分隔列及重复框架对每个分隔列允许执行一些动作。根据所想用的顺序显示数据，可以将分隔顺序属性设为递增的或递减的。如果将分隔顺序设为“无”，会抹掉数据列的分隔特征 —— 这是记录组的通常设置方法(见图7-6)。

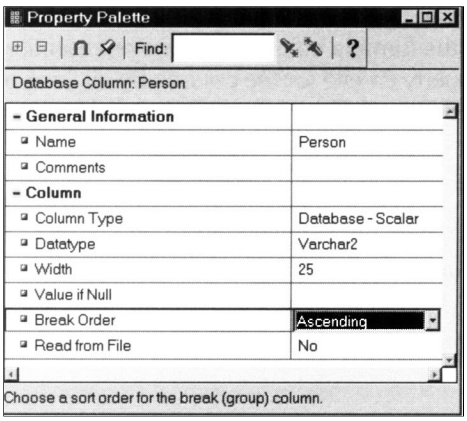


图7-6 记录组的设置

把汇总列放进分组中来表示想要进行汇总的那个级别。如果想对分组记录求和，就把它放进分隔组的分隔列。如果要对报表中所有记录求和，像第4章中一样，要把汇总列放在所有分组的外面。如果大多数要建立汇总列，可以像第 4章的Ledger汇总报表一样使用 Report Wizard。对于向导不能实现的特殊分隔组的汇总列，首先创建分隔组，然后单击Data Model Editor工具板上的汇总列工具，然后单击分隔组，双击所产生的结果列，显示汇总列的属性板(见图7-7)。

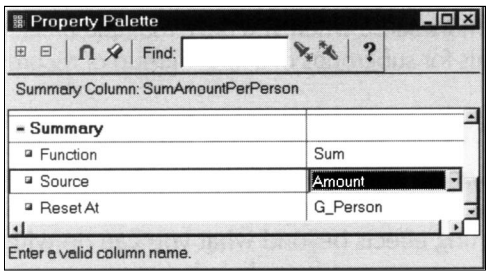


图7-7 汇总列的Property Palette

函数属性允许设置要完成的汇总的种类。可以选择求和 (sum)、平均值(average)、最小值

China-pub.com 下载

(minimum)、最大值(maximum)、计数(count)、第一个值(first value)、最后一个值(last value)、百分比(percent of total)、标准方差(standard deviation)或不同值(variance)。

Source(数据源)属性有一个下拉列表，可以用来选择汇总的数据源。它能自动显示在分隔组下面记录组的所有合适的列，在设置 LedgerGroup组的Subtotal列时，它不仅显示Amount列而且显示Quantity及Rate列。如果查看在任何分组外的总合计列，将会显示所有可能的列：Amount、Rate、Quantity及Subtotal。也可以在Amount列或Subtotal列的基础上求总合计，结果是一样的。

Reset At(复位)属性告诉Developer从哪里开始汇总。考虑对分隔组分类汇总的情况，希望对分组中的所有行的求和进行分类汇总。 Reset At属性指定分隔组，并告诉Oracle Developer在开始下一分组之前重新把值设为零。对于分隔组的汇总列，复位点可以是分隔组、页面及报表。如果选择分隔组 (Ledger汇总报表中是G_person)，Oracle Developer从分隔组进程的第一条记录开始计算值，然后在继续下一分组之前复位该值。这样结果是对每一个分组进程产生一个和。如果选择页面，就在每页前复位该值。分组可以是跨页的或在同一页中出现多分组数据。如果出于某种原因要计算与页相关的值时，通常选用页面设置。如果选择报表，Oracle Developer在开始产生报表前把该值设置成零，并且以后不重置。对每一个分组进程产生一个运行总和。而运行总和 (running total)是在整个报表内求和，而不是在每分组记录开始时求和。如果把汇总列的复位属性设成报表，可以在分隔组之后打印结果。

Compute At(计算)属性告诉Developer从哪里计算某个集合的统计百分比函数值。可以分别在报表、页面或分隔组的级别上进行，即允许 Developer在不同的总和上计算百分数。报表级意味着占报表总数的百分数，页面级意味着占页面总和的百分数，分组级表示分组总和上的百分比。当有多个分隔组，每一大组记录中有多个小组，并且要打印占小组的总和百分比时，最后一种选择是非常有意义的。

7.1.4 多重查询及数据链接

有的报表一个查询足够了，但有时也需要多个查询。 Oracle Developer使用多个查询可以制做出非常复杂的报表，这是单个查询所不能达到的效果，包括分隔组、汇总及公式列。

一个典型的例子是主 -从报表。第4章介绍了如何开发技能数据库的主 -从表单，即让每个工人与其技能相关联。如何开发一个类似的报表呢？

Oracle Developer Reports使用一种非常类似于表单对象的结构来完成同样的事情。它不是使用两个数据块，而是在数据模型中有两个查询。在主数据块不使用关系对象，而是有一个数据链接。数据链接 (data link)正像所称呼的一样：是链接两个数据集的一种方法。实际上是有点复杂度的：在一个主 -从报表中真正做的是从主分组链接至从分组查询，来设置查询的WHERE子句。

1) 要创建一个新报表，可以用 Report Wizard定义一个查询。要从数据库中查询工人的SQL语句如下：

```
SELECT Name
  FROM Person
 WHERE Lodging IS NOT NULL
 ORDER BY 1
```

2) 在Data Model Editor中显示查询及它的分组。

China-pub.com
下载

3) 单击Query工具，然后单击Data Model Editor中Worker查询旁边的画布。

4) 输入下列SQL:

```
SELECT Name, Skill, Ability
  FROM WorkerHasSkill
 ORDER BY 1
```

注意，在新分组中的Name数据库列是以Name1名字形式出现的，以保证名字唯一(见图7-8)。

5) 单击Data Link工具，然后单击并拖拉Worker Name列至Skill Name1列。当松开鼠标时，链接把查询及查询名字下的列名字联系在一起。

如果现在返回到Object Navigator并且查看Data Link标题下方，会发现新的数据链接(见图7-9)。

如果双击Data Model Editor中的数据链接线，或Object Navigator中的Data Link图标，会看到该对象的Property Palette。可以设置链接的两个不同属性(见图7-10)：

- SQL Clause：HAVING，START WITH或WHERE。HAVING是基于GROUP BY的选择；START WITH实现CONNECT BY；WHERE是一种标准选择。
- Condition：是用于两列间的关系操作符。

如果在表之间已定义了关键字的约束，可以直接链接两个查询，而不用引用各分组及列。在Worker-Skill(工人-技能)链接的例子中，单击数据链接工具，然后单击WorkerQuery并拖拉到SkillQuery也会得到如前面一样的结果。在数据字典中查找各约束之后，Data Model Editor在WorkerGroup的Name列与SkillQuery的Name1列之间建立链接。

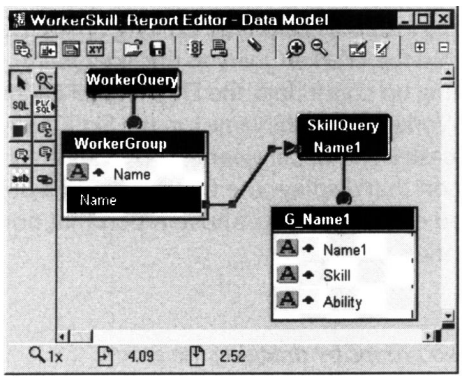


图7-8 Name数据列

图7-9 新的数据链接

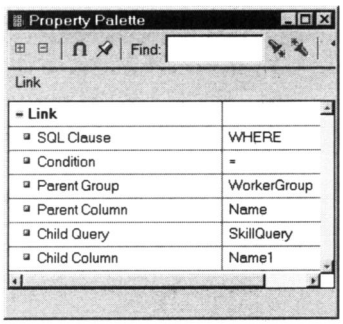


图7-10 设置链接属性

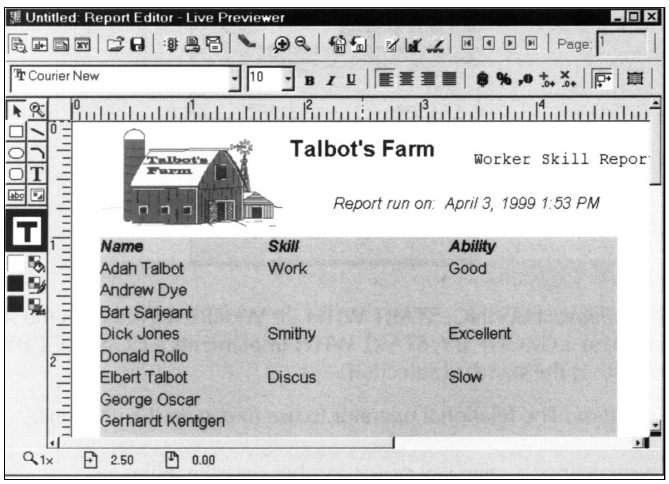


图7-11 WorkerSkill主-从报表

China-pub.com 下载

在链接了主-从查询之后，可以用Report Wizard定义一个Group Left(左分组)的报表。以分隔组的模式显示的两组数据，如图 7-11所示。也可以定义一个Group Above(上分组)报表，在每个工人的记录下面打印一系列Skill的记录。

注意 通过拖拉分组名到其他分组名的方法，也可建立两个组的链接。在这种情况下，主分组的名字在主分组中以一个列的形式出现，从分组的名字出现在从查询中。Property Palette中的列属性是空的，这种链接变化情况是很少见的。在从查询中可以引用主列作为约束变量使用，一种用途就是实现基于值的查找。

有了数据链接，就完全具备了创建各种报表数据模型的各种工具。下一节介绍如何使用数据模型的产生结果，如何按照要求的格式进行设计。

7.2 格式化报表

第4章介绍了格式化报表的基本知识，这一节介绍 Oracle Developer Reports中一些可用的格式化工具。Report Builder中最重要的接口是Live Previewer，它可以显示实际的报表，也提供了直接格式化报表的方法。利用 Live Previewer，可以非常容易地格式化报表的布局及提示文本，尤其是报表的标题及边界。

提示 通过Live Previewer或通过属性可以看到实际报表的所有内容。有时可能需要对框架实行一些更精细的控制，为了实现这些，应当学会使用Layout Editor，这是一种显示设计格式而不是显示报表的实际视图的老式接口工具。在大多数情况下， Layout Editor比Live Previewer难以使用的，因此这里不详细介绍其内容。

7.2.1 Live Previewer

在Live Previewer中，对于每件要作的事情基本上就是选择一个或多个域。在Live Previewer中看到的每个对象实际上是一个域。单击某个对象，会看到属于该域的所有对象都出现手柄(handle)。如果该域不在一个重复域中，只会看到单个域带有手柄。如果框架是重复的，那么会看到所有重复的域带有手柄，如图7-12。

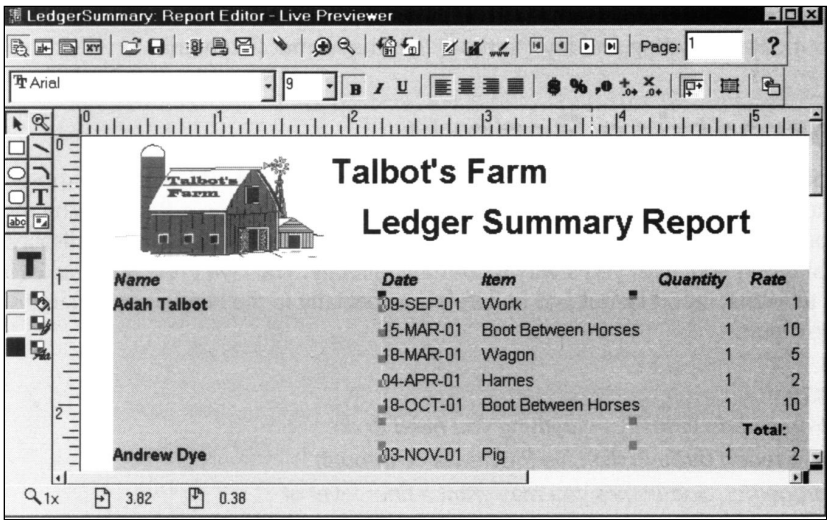


图7-12 Live Previewer中的Ledger Summary报表

China-pub.com
下载

按住SHIFT并单击附加域或框架可以进行多重选择。可以单独地选择各标题的不同的格式及尺寸。用Edit | Select Parent Frame 菜单项或Select Parent Frame工具，可以选择围绕各域的框架。如果双击一个域，则会看到它的 Property Palette。

如果使用某个格式化工具，则可以格式化所选择的各个对象。如果未选择任何对象，则会为报表将来加入的对象设置缺省的格式，关于所能完成功能的详细介绍参看后一节“格式化工具”。

利用各手柄可以完成三种特定功能：

- 可以拖动角部或底部或顶部手柄来改变字体大小。
- 可以拖动边手柄改变域或框架大小而不改变字体大小。
- 可以拖动域的中部把域向左或向右移动。

当改变域的大小或移动域时，对其他域及标题的影响取决于 Flex Mode(伸缩方式)的设置。工具条上的Flex Mode按钮显示这种设置情况。如果 Flex Mode是关闭的，改变大小或移动域时对其他域或标题没有影响，可以把域移到任何地方，甚至与其他域交错，页头不跟着移动。如果Flex Mode是打开的，当改变一个域的大小或移动时，如使它变大或移动到右边时，则其他域也跟着移动以适合变化的大小或域的位置。如果向右移动域或改变域的大小，而在页面上域的右边又没有足够空间来移动各域时，鼠标符号变成一个“不允许”符号，当释放鼠标时则会取消移动或改变大小。

注意 必需仔细地观察拖动限制，否则利用Oracle Developer无法完成重新格式化布局。这个过程就像移动小方框的那种游戏，推移各片以允许其他各片的移动。通过移动其他几个域或重设大小，为重设另外的域提供移动的空间。

工具条上的四个页面工具(第一页、前一页、下一页及最后一页)，允许在Live Previewer中逐页地检查报表的各部分。在Live Previewer中看到的内容与所打印出来的报表是完全一样的。必须查看报表的所有页，检查诸如报表总计、表头页的图表等等(见图7-13)。

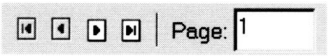


图7-13 顺面工具

提示 当开发报表时，在测试数据库中放置足够多的数据，使得在Live Previewer中显示多页报表，这是一种好的办法。否则，会发现最终结果可能是不满意的。另一方面，在测试数据库中有足够多的数据，可以产生很多结构一致的报表页，并强制滚动很多页才能看到最终内容。在Page域中输入页号并按回车可以转到一特定的页，只要大约知道要到哪一页即可。

7.2.2 修改报表

要修改报表，可以用 Live Previewer完成大多数格式化任务。如果要做较大的改变，可以使用Report Wizard重新设计基本布局。用 Property Palette可以改变某些重要的属性，更精确地控制报表的格式化。

1. 使用Report Wizard重格式化报表

第四章演示了用Report Wizard构造基本汇总报表的方法。 Report Wizard不仅可以控制布局中的最重要的部分，而且允许重新设计布局，从而保留已做的格式化修改。可以用任何编

China-pub.com 下载

辑器中工具条上的Report Wizard工具运行Report Wizard，或使用Tools | Report Wizard菜单项。

注意 在Report Wizard中选择的报表布局模式对于布局选项有主要影响。例如，如果选择Group Left模式，Wizard则增加标签页来定义分组的列。由于某些模式不使用标签页，某些使用Wizard的新手容易被这些选项搞混。

2. 框架属性

框架(Frame)是报表中包含分组对象的一个矩形区域。框架是报表中的对象，它通过对各个域及其他框架分组来组织报表布局。框架把报表分成几个部分。框架可以设置相对其他框架的相对位置。重复框架 (repeating frame)为Developer在数据模型中设计的分组定义循环结构。

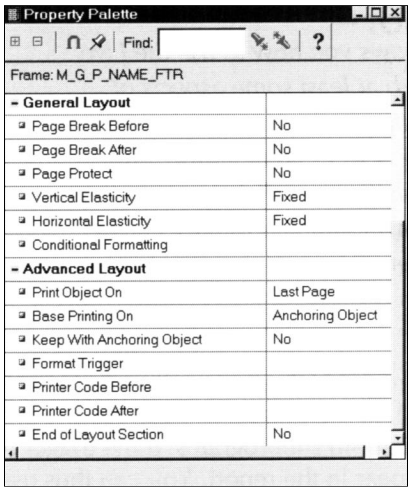


图7-14 Property Palette中的布局特性

框架是出现在对象导航及 Live Previewer中的对象。有时，一个报表尤其是复杂的报表，看起来大多数对象都是框架。框架在 Live Previewer中以虚线的形式出现。框架对象属性板中包含一组布局特性，并分成 General Layout (通用布局)和Advanced Layout(高级布局)两个属性组(见图7-14)。

页面属性规定 Oracle Developer如何格式化框架。Page Break Before和Page Break After规定报表如何相对框架设置页标记。例如，要把分隔报表的每组数据放在一单独的页上，可以把该分组的重复框架的 Page Break Before或Page Break After设成Yes。Page Protect就像是字处理器中的Keep Together属性，它告诉Oracle Developer，如果可能的话把框架中的所有对象放在同一页上。 Keep With Anchoring Object属性就像在字处理器中的Keep with Next特征。如果框架及它的锚对象 (anchoring object)(通常是另一个对象)不能放在同一逻辑页上，Oracle Developer把它们移到下一逻辑页上。

Vertical和Horizontal Elasticity说明框架如何随着数据扩展，参看下一节“伸缩性及大小可变性”。

Conditional Formatting允许在框架对象中应用格式异常。根据框架中基于数据元素的条件，使用格式异常可以隐藏框架或改变它的填充及边界颜色。

Print Object On及Base Printing On属性说明框架包容的或附靠的对象的打印关系。利用这个属性可以修改框架的重复打印特性。例如，假设只要在第一页上打印标题，可以把附靠对象的Print Object On属性设成First Page。

注意 该例子介绍如何用Live Previewer处理框架的某些方面内容。选择Live Previewer的一个标题，然后单击Select Parent Frame工具，选择Tools | Property Palette菜单项，现在可以设置属性板中的各属性，并且Oracle Developer会在Live Previewer重显示报表。

Format Trigger属性说明一个PL/SQL布尔函数(返回True或False的函数)。在触发器中，可以设置可通过内部子程序访问的任何属性。如果触发器返回 False，框架及它的内容都不会出现在报表中。因此，可以用附加的框架及它们的触发器来格式化或压缩报表的特定区域。例如，如果只对某些固定的分组显示一组特殊域，则可以对这些域加一个框架并对它增加一个

China-pub.com
下载

触发器，来检查当前分组信息并返回适当的布尔值。

重复框架在属性板中有两个附加部分 (见图7-15)。

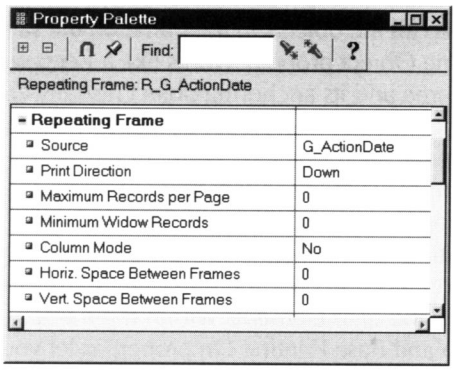


图7-15 重复框架在属性板中有两个附加部分

框架的Source(数据源)是框架获取数据的分组名。Print Direction属性告诉Oracle Developer向下(表格式报表的通用格式)或横向或两者组合重复打印各记录。以横向/向下方式打印，表示Oracle Developer先横向打印各记录，直到不能打印一条记录为止，然后在下一行打印该记录并继续下去。向下 /横向打印以竖排形式打印，记录从上往下一直打到页底部，然后从页顶部打印下一竖排，直到右边界为止。 Column Mode属性通过把一些连续的记录放在一些顺序页上来控制组合的打印方向，如果有必要可以把这些页剪切并粘贴在一起。在Layout Editor中，打印方向用框架边界上的小箭头表示，如7-16图所示。要清楚了解它们的含义，需要试验这些设置，观看所打印出的报表的样式。

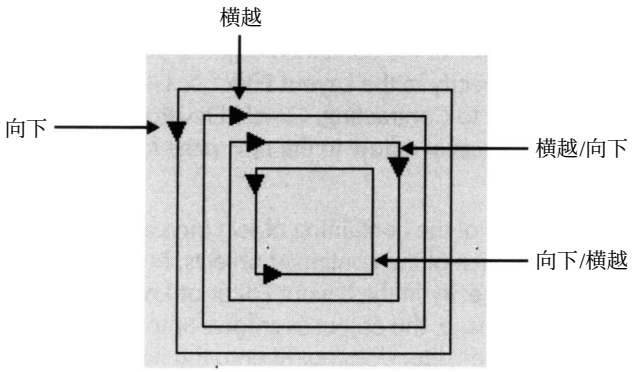


图7-16 在Layout Editor中指示打印方向

Maximum Records per Page(每页最大记录数)限制Oracle Developer在一页上打印的记录数。例如，可以利用此属性，在每页上打印一个记录。

Minimum Window Records (最少窗口记录) 规定记录的最少数。如果页面不能包含所规定数目的记录， Oracle Developer把记录移到下一页。这类似字处理器中的窗口控制，它规定在每页的底部每段至少要有两行，不能只是一行。

Horizontal or Vertical Space Between Frames(在框架间水平或垂直间距)可以给每一记录加一边界，即在重复的记录之间打印规定数量的空格。

最后，Filter Type属性与其在分组中所对应的属性扮演相同的角色。可以在框架的基础上管理记录的过滤，而不用在分组的基础上。如果过滤记录，应当首先在分组上进行，除非有多个框架引用该分组，并且有不同的过滤要求，这种情况是很少出现的。

7.2.3 伸缩性及大小调整

Horizontal Elasticity and Vertical Elasticity属性描述一个对象在运行时如何调整以适合它

China-pub.com 下载

所包容的对象及数据。不同种类对象的伸缩性显示不同的效果。框架大小的变化取决于框架内容。文本对象随文本大小而变化，图像及图形对象会按比例缩放以适合包容它的对象的大小。

共有四种类型的伸缩性：

- Fixed(固定的)：包容对象的大小不随所包容对象的大小变化，并且大小不会超过在Layout Editor或Live Previewer中说明的大小，如果超过则截断所包容对象。
- Contracting(缩减)：包容对象的大小可以垂直地或水平地缩减，以适合缩包容对象的大小，但包容对象的大小不会超过在 Layout Editor或Live Previewer中说明的大小。如果设置水平伸缩性为缩减的， Oracle Developer可能会截断所包容对象，但如果垂直伸缩性为缩减的，对象会溢出到下一页上。
- Expanding(扩展的)：包容对象的大小随所包容对象的大小会垂直或水平地增加，但大小不能少于在Layout Editor或Live Previewer说明的大小。如果对象不能放在一页上，则对象会溢出到下一页上。如果编辑器上对象的大小不适合在一页上，对象会移到下一页上，如果有必要还会溢出。
- Variable(可变的)：包容对象的大小可以扩展或缩减以适合所包容对象的大小，在Layout Editor或Live Previewer上所包容对象的大小与运行时的大小没有什么关系。如果对象不是恰好在一页上，对象会溢出到下一页。

因为可以独立地设置垂直或水平伸缩性，所以伸缩性类型有 16种可能的组合。例如，对一个文本域，水平伸缩性设为固定的，垂直伸缩性设为扩展的，文本则可以维持相同的宽度，但当文本超出编辑器中说明的大小时会向下扩展。如果文本超出当前页，该域会扩展到下一页上。再如，设置Horizontal Elasticity是的，报表显示记录各域向右互相挨着，而不是把各域线性排成一列。

可对Ledger报表进行设置实验，观察对一个文本域设置选项的不同组合的不同效果。例如，一个项目可以折成多行。通过指定域的值及选择手柄可以立即看到效果。

注意 溢出打印的规则是极为复杂的，详细内容参看报表联机文本。

如果各域的值是从计算数据源得到的，如页数等，并且对各域使用了除固定伸缩性之外的任何设置，某种时候可能会遇到错误信息。当数据源直到整个报表格式化之后才能提供值时，会出现这种问题。这意味着 Oracle Developer在不能准确地知道该域包含什么数据的情况下对其格式化，这将会产生错误。这样的域必须具有水平及垂直固定的伸缩性，如果需要，Oracle Developer会转换成这种设置。在如下的各种情况会发生这样问题：

Source：数据源是总逻辑页数、总物理页数、总版数、逻辑页号、物理页号或版号之一。

汇总或公式列：汇总或公式列的 Reset At或Compute At属性设为页面，或者依赖于这样的列。

7.2.4 格式化工具

在Live Previewer中不仅可以立即看到格式化的结果，并且在工具条中提供了方便使用的大部分格式化工具(见图7-17)。

放大镜图标是放大及缩小工具，允许放大或减小 Editor的放大倍数，以便看到细小的区别或较大的图片。下两个图标插入日期 /时间及页号，与三个向导 (报表、图表及 Web)中的一样。

China-pub.com
下载

工具条的第二行说明字体、字体大小以及黑体、斜体及下划线设置等，用不着去操作 Format | Font菜单项。下面三个图标控制所选择对象的调整 (左对齐、居中、右对齐)。还有几个图标允许改变数字域的特征：插入货币符号、百分号及千位标识符，或者插入或移动数字的小数位置。所有这些工具在 Format菜单中有对应的菜单项，但使用工具条更方便一些。只需选择需格式化的项目，然后单击工具图标即可。

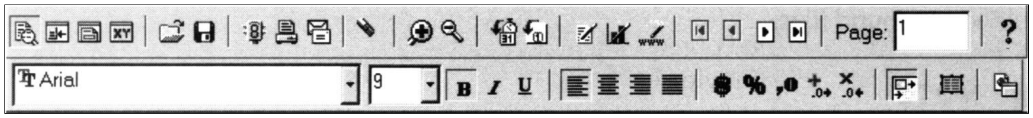


图7-17 工具条中的格式化工具

Format菜单的其他格式化选项允许更准确地控制报表布局：

- 文本选项：行间距及打印方向。
- 线条选项：宽度、边界的凹凸显示、虚线及箭头效果。
- 图形选项：各种画图选项，如图像灰度或圆角边效果。

可以使用Format | Reduce Image Resolution (格式｜减小图像分辨率)控制存贮在报表中图像数据的数量。当减小图像的尺寸时，从减小分辨率版本的图像中得到的图像，与从原始分辨率图像中得到的图像质量一样好。减小分辨率可以节省大量空间。应当对报表进行试验，以确定在特定情况下最有效的分辨率 /空间划分。

7.2.5 按钮及交互式报表

Oracle Developer Report不局限于静态的、可打印的报表。在布局中可以增加按钮，在运行时用户可按下按扭执行多媒体或 PL/SQL代码。在按钮中也可以加入任何想加入的代码。通常的用法是提供下拉报表 —— 对某个记录的明细记录产生第二个报表。也可以播放视频或声音文件。

注意 当按钮用于有趣及有用的报表时，当然混淆了表单及报表应用程序的界线。开发主动式应用程序，使用Oracle Developer Forms环境比使用报表更容易，且更方便。当打算产生一个联机报表(如在因特网上)而不是打印它时，使用按钮似乎比较合理。

1) 创建主报表及下拉报表。创建一个主报表，列出 Ledger中每个人的费用总数。然后创建一个下拉报表列出每个人的每件事务，并带有汇总。数据模型中，第一个报表 Person.RDF的SELECT语句按人员分组：

```
SELECT p.Name, SUM (Quantity*Rate) "Total Amount"
 FROM Ledger l, Person p
 WHERE l.Person = REF(p)
 GROUP BY p.Name
```

该查询连接Ledger和Person两个表，它通过连接 Person表的对象引用及 Person表上的REF函数产生的引用来实现。该连接允许引用 Person表的名字，但它不是Ledger表中的列。

对于第二个报表(Person Detail.RDF)，数据模型的SELECT选择与一个特定人员有关的数据，它通过约束变量实现：

```
SELECT Action, Item, Quantity, QuantityType,
```

China-pub.com 下载

```
        Rate, Quantity * Rate "Amount"
  FROM Ledger l, Person p
 WHERE l.Person = REF(p) AND
        p.Name = :P_Person
 ORDER BY 1
```

当在Report Wizard或Data Model Editor中创建数据模型时，报表自动创建用户参数对象并命名为P_Person。可以把这个参数放进参数表单中，通过对象的列表值属性，使用一个SELECT语句可以为对象提供一个选择列表值，如：

```
SELECT Name
  FROM Person
 ORDER BY 1
```

注意 当通过Person报表访问PersonDetail报表时，实际上不用显示参数表单，所以不用设置参数表单或列表值。采用这种方法可以增强 PersonDetail报表的可重用性。当然，也可能希望在报表头的某个地方显示参数值，如在顶部边界等。

2) 继续进行Report Wizard中的标准格式化步骤，直到两个报表都设好以后。

3) 在Person报表中加入按钮。打开 Layout Editor，选择Amount域，并重置大小以留出空间来紧靠着放置按钮。只要按钮在 Person的重复框架里面，对于每个记录它都会重复出现。

4) 单击工具板上的Button工具，然后在框架中画一个矩形按钮。

5) 用Format | Bevel把形状设为凸起的(Raised)(见图7-18)。

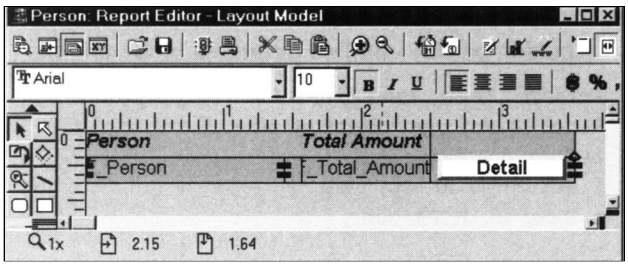


图7-18 形状为凸起的

6) 双击按钮显示它的Property Palette(见图7-19)。

7) 输入标号文本，如“ Details ”。

8) 检查Live Previewer或Layout Editor中的字体格式，并用合适的工具改正它们。

9) 回到按钮的 Property Palette，把 Button Behavior的Type属性设为PL/SQL。

10) 单击PL/SQL Trigger属性，输入如下的按钮触发器的PL/SQL代码：

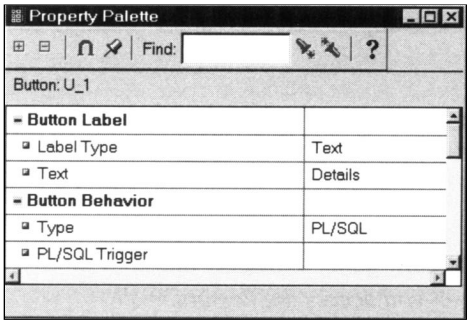


图7-19 显示Property Palette

```
PROCEDURE U_1ButtonAction IS
BEGIN
  SRW.Run_Report(
    'PersonDetail.rdf P_Person="'||:Name ||'"');
END;
```

PL/SQLProgram Unit Editor支持PROCEDURE模板，所需完成的是向Run_Report输入一个过程调用。在命令行中需传递一个参数，该参数取值于 Person表中当前记录的列。

China-pub.com
下载

11) 编译触发器，准备试运行报表。

当Live Previewer在最顶层窗口时，使用View | Runtime Preview菜单项可以用刷新的数据运行报表。不能直接单击Live Previewer中的按钮。图7-20显示报表Previewer的运行结果。如果单击某个person的Detail按钮，将会看到如图7-21的明细报表。

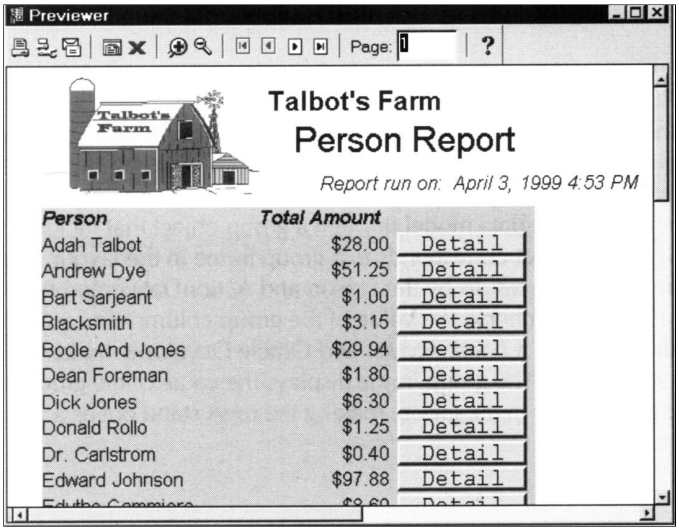


图7-20 Runtime Previewer中的Person报表

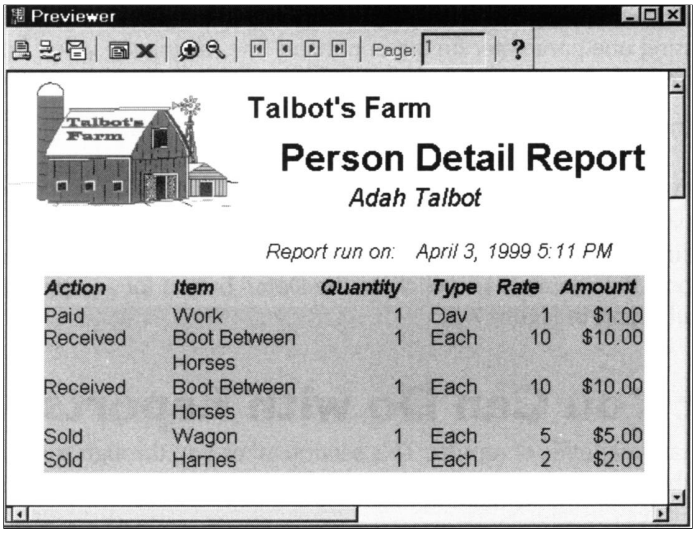


图7-21 Runtime Previewer中的PersonDetail报表

7.3 可以用报表做什么

实际应用中会有各种各样的报表。这一节介绍一些基本的变化，并介绍如何利用前节中介绍的工具得到期望的效果。

7.3.1 表格式报表及分组报表

从前面章节“建立报表数据模型”知道，一个分组是用一列或多列的唯一值分开的一组

China-pub.com 下载

数据记录行。例如，第4章的Ledger Summary报表中，按Person列对各数据行分组。在数据模型中，用包容分组列的对象表示一个分组。在布局中分组对象对应一个分组框架。数据模型中的查询用Person及ActionDate列来对结果数据进行排序。打印报表时， Oracle Developer对打印的每一行检查分组列的值。当值改变时，出现一个分隔 (break)，并且， Oracle Developer为下一个分组建立分组框架。对于一个分组中所有行，分组框架只显示一次分组列的值，使报表中同一分组的记录能清楚地表示出来。

在Report Wizard中建立表格式报表时共有三种模式：

- Tabular(表格式)：由各列组成的简单的表格，各数据行不分组。
- Group Left:(左分组式)：有一个或多个分组列的表格式报表，分组框架显示在被分组的数据的左面。
- Group Above(上分组式)：有一个或多个分组列的表格式报表，分组框架显示在被分组的数据的上面。

用Report Wizard的样式(style)屏幕可以为报表选择某一种风格。根据选择的风格类型，Report Wizard显示不同的屏幕(当创建报表时)或标签页(当修改已创建好的报表时)。以上三种模式都显示如下屏幕或标签页：

- Style(样式)：报表的样式(表格表、左分组或上分组单选钮)。
- Data(数据)：定义产生报表数据的SQL SELCET语句。
- Fields(域)：显示在报表中的列。
- Totals(合计)：聚集到分组总计或者报表总计中的列，以及在进行该工作时所使用的汇总函数。
- Labels(标签)：报表各域的标签及大小。
- Template(模板)： 用于设计报表的模板。

图7-22显示以表格式模式打印第4章的Ledger Summary报表的结果。报表的最后一页有一行表示对整个报表的Amount列求总计的结果，在图上没有显示出来。在分隔报表中，报表会自动按分隔列对各行排序，但在表格式报表中则不会。在这个例子中， Name列是引用Person

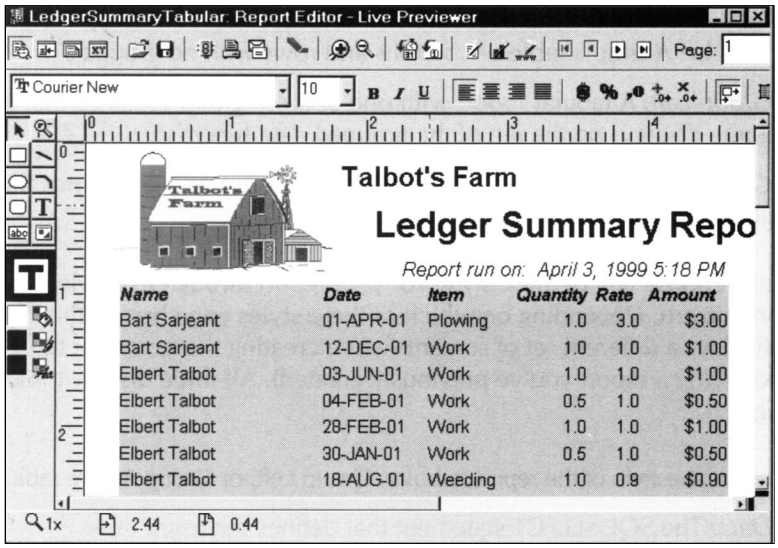


图7-22 表格式Ledger报表

China-pub.com
下载

表的一部分，无法按名字对输出结果排序，因为在 ORDER BY语句中不能引用Name列。

左分组及上分组还有另一个屏幕/标签页：

■ Groups：用来分组数据的列。

图7-23是左分组式的Ledger Summary报表，图7-24是它的上分组式的报表。

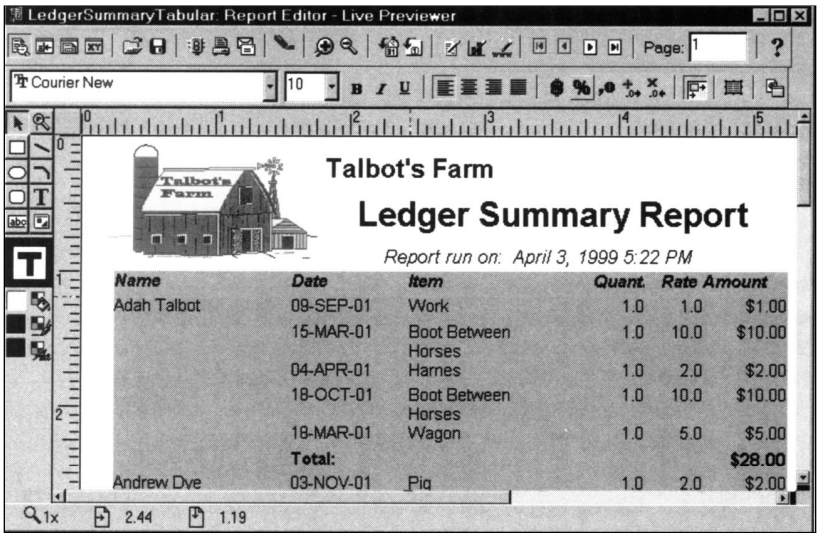


图7-23 左分组式Ledger Summary报表

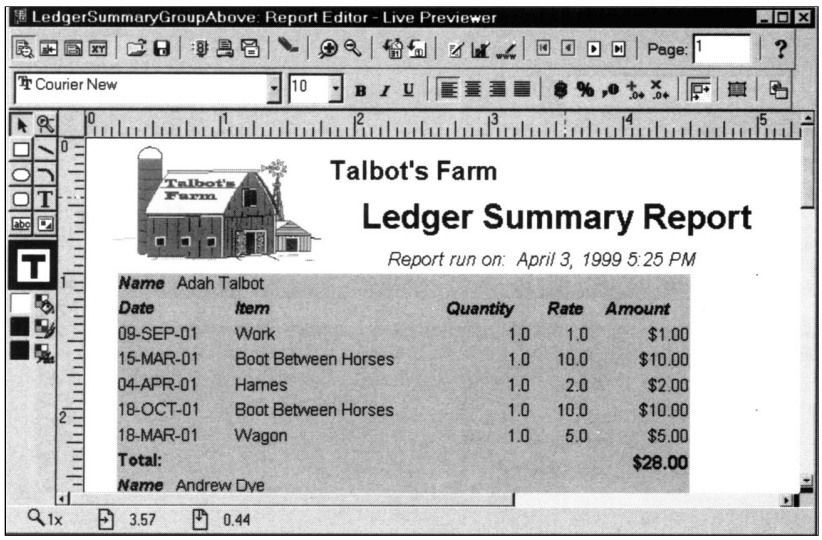


图7-24 上分组式Ledger Summary报表

三种报表都包括汇总数据。在图中可以看到 Report Wizard为每个组产生了小计，并为分组报表产生了报表总计。

7.3.2 表单报表

表单报表是一种一次只显示一条记录的报表，非常像设计的表单。用表单报表可打印类似服务订单、邮递单及其他需要每页一条记录的项目。作为一个例子，考虑一个报表需要打

China-pub.com 下载

印Lodging表(住所表)中的寓所信息，每个寓所用一组域表示：

1) 为建立新报表，起动Report Wizard，并选择表单模式。

2) 输入下列查询：

```
SELECT Lodging, LongName "Full Name", Manager, Address
  FROM Lodging
 ORDER BY 1
```

3) 其他屏幕内容不需改变，只单击Finish即可。

这样就建立了一个每页有一条记录的表单报表，自左至右排放各个域及它们的标题，当有溢出时卷到下一行。在这个例子中，每页一条记录只有很少的数据显示。

4) 选择框架对象(只有一个)并把每页最大记录数(Maximum Records Per Page)属性从缺省值1变成0，这表示在一个页面上显示尽可能多的记录。

5) 把Lodging(住所)域改成黑体显示，以强调每个寓所。

图7-25是该报表的结果显示。

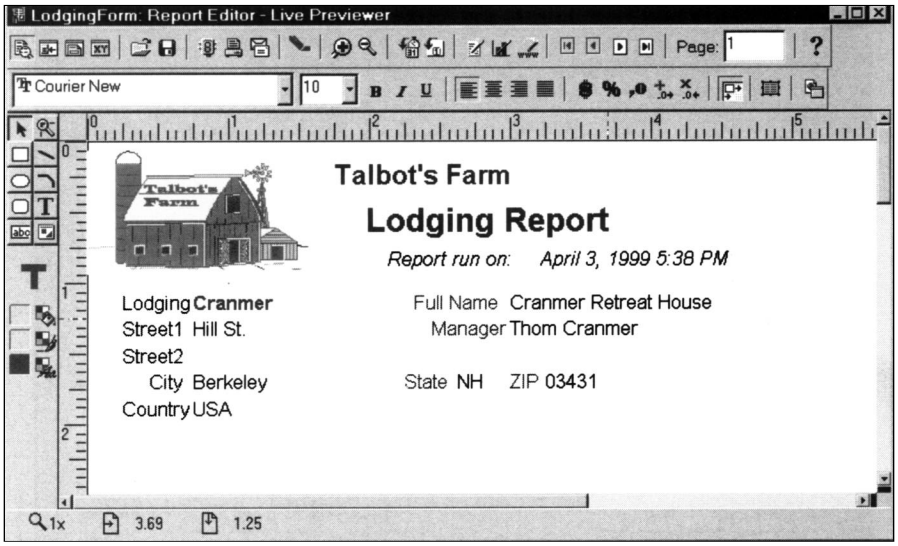


图7-25 住所表的表单报表

7.3.3 邮签报表

Talbot农场的商业公司可以用邮签为某个人员分组创建邮件。邮签报表打印类似标签的通知单。例如，Talbot农场用这种报表，可以产生支票标签或发给各雇员的邮件。

邮签报表打印重复记录时，在进入下一页的中间的大块空白内不带任何标题。当打印时，通常是打印在特殊的带胶的邮签纸上，打印完后可以揭下标签使用。这一节介绍如何快速产生这样报表的简单原型。

1) 用Report Wizard建立新报表这个报表的查询比前面例子的查询稍微复杂一点，因为它涉及到两个连接的表，而不是只一个表。查询中有 Person表中的名字列及Lodging表中人员寓所的LongName和Address列。两个表的连接是通过产生 Lodging数据行的引用，然后与 Person表中Lodging对象引用相连接实现的。SQL按照人员名字排序结果数据：

```
SELECT p.Name, l.LongName, l.Address
```

China-pub.com
下载


```
 FROM Person p, Lodging l
WHERE p.Lodging = REF(l)
ORDER BY 1
```

下一屏幕(见图7-26)类似于其他报表的 Fields屏幕，但对于邮签结构有其特殊的内容。Text屏幕可以定位邮签原型的各个域。它提供了不同符号按钮，用于分隔邮签的各项目 (新行、空格、逗号、破折号及句号)。

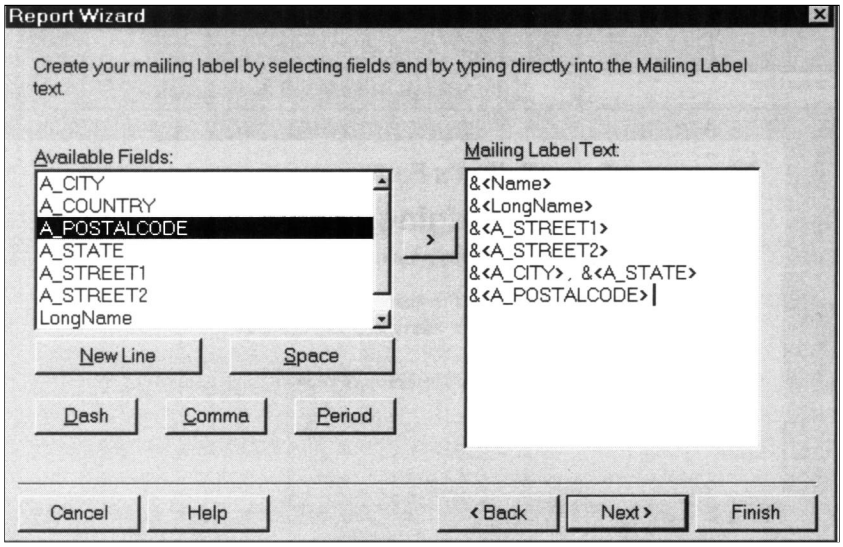


图7-26 Talbot邮签报表的Report Wizard Text标签页

2) 为了创建Talbot邮签选择Name，单击>按钮，单击NewLine按钮，类似地重复加入Long Name，A_Street1及A_Street2域。然后，增加 A_City，单击Comma、Space，再接着增加A_State、Space、Space，最后增加A_PostalCode域。最后一个域折到一新行，但它在邮签上却会出现在同一行上。

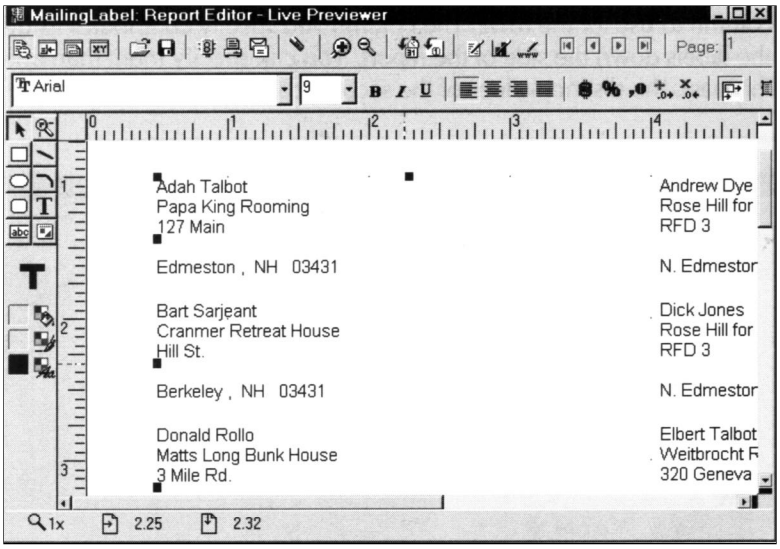


图7-27 Talbot工人的邮签报表

China-pub.com 下载

在Wizard中其他还可用的屏幕是模板屏幕。如果有模板可用于邮签报表，可以使用该屏幕。

3) 单击Next，选择Draft模板或不选择模板。

提示 为标准邮签布局建立模板的方法是非常好的。否则，会花大量的时间用 Live Previewer及Property Palette来设置边界、空行数及准确地定位。

图7-27显示了一些基本布局工作的结果。

7.3.4 矩阵报表

有时有两种类型的对象，希望对两种对象的每一种可能组合，均给出它的汇总值，这时就需要使用矩阵报表(matrix report)。矩阵在统计界称做交叉列表或十字报表，在一个矩阵中关联到两个或多个变量。一个变量的可能值沿顶部排放，另外一个变量的可能取值沿侧面向下排放。

Oracle Developer Report支持矩阵报表，它采用一个单独的报表模式，并给出可以控制报表的布局一些选项。例如，假设Talbot农场的一个行政官，希望看到一年内每星期的现金在工人间的流通情况。使用矩阵报表，用每周作为列，每个工人作为行，可以显示每个工人每周的费用数量。用Report Wizard创建一新报表，在模式页上设置报表模式为矩阵。该设置激活了Wizard屏幕允许设置行和列结构，及单元的结构。

首先在Data屏幕输入查询设置基本的数据模型：

```
SELECT To_Char(l.ActionDate, 'MM')||'-
          '||To_Char(l.ActionDate, 'Mon') "Month",
       p.Name, l.Quantity*l.Rate "Amount"
  FROM Ledger l, Person p
 WHERE l.Person = REF(p) AND
       p.Lodging IS NOT NULL AND
       l.Action = 'Paid' AND
       To_Char(l.ActionDate, 'YYYY') = '1901'
```

该查询检索三组值：月份、工人名字及工人的薪金总计。月份由数字月份 (01，02等)及三字符月份名字(‘Jan’,‘Feb’等)连接而成。增加月份数字值的原因，是为了矩阵报表用唯一的值排序，如果只有字符名字，月份将是以字母顺序而不是以月份顺序显示。

该查询连接到Person表来限制对工人的查询，只查询有住所的人员(住所列非空——Lodging Is NOT NULL)。REF连接(参照连接)在Ledger表中基于对Person对象参照引用的表。活动为‘Paid’且该查询通过YYYY日期格式限制总帐项到1901年。

Report Wizard的下一屏幕(图7-28)询问采用数据库的哪一列做行。用被指定的列的不同值作为标志，在报表的一边向下分布。也可以说明多个列，这时第二列的不同值嵌套在第一列的值中间，在多次重复的结构中依次类推，如各个月中各星期；人员的活动类型等。例如，如果要显示每个人购买、销售、支付或收入的总和，可以用人员作为一级行变量，活动类型作为二级行变量。报表单元将会为人员和活动的每个组合产生一个求和值。

下一屏幕(图7-29)说明哪个数据列用做报表列，并分布该列的不同值作为列标题。这里也可以说明多个级别。

注意 下一屏幕让你建立单元的结构。通常使用标准汇总功能之一如Sum来计算每个单元的特殊列的总和值，如图7-30所示。该例在工人和确认单元的目的查询中计算行的

China-pub.com
下载

总和值。如果无这样的行，单元中将无值。

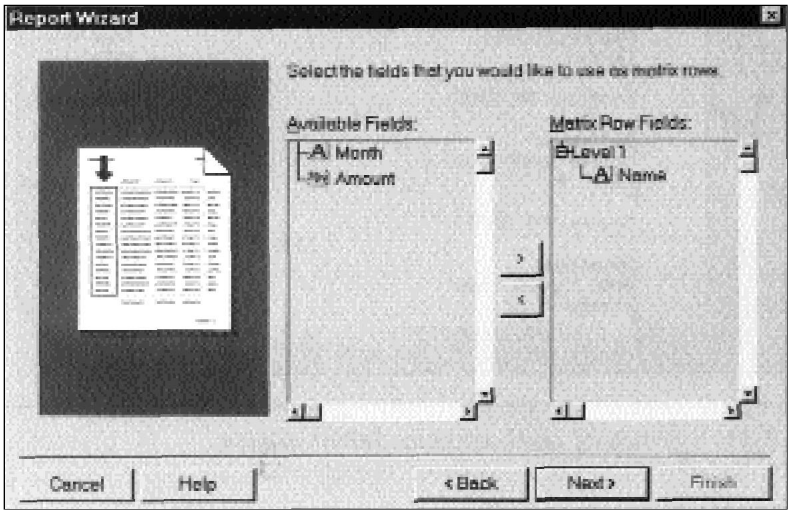


图7-28 矩阵报表的Report Wizard Row屏幕

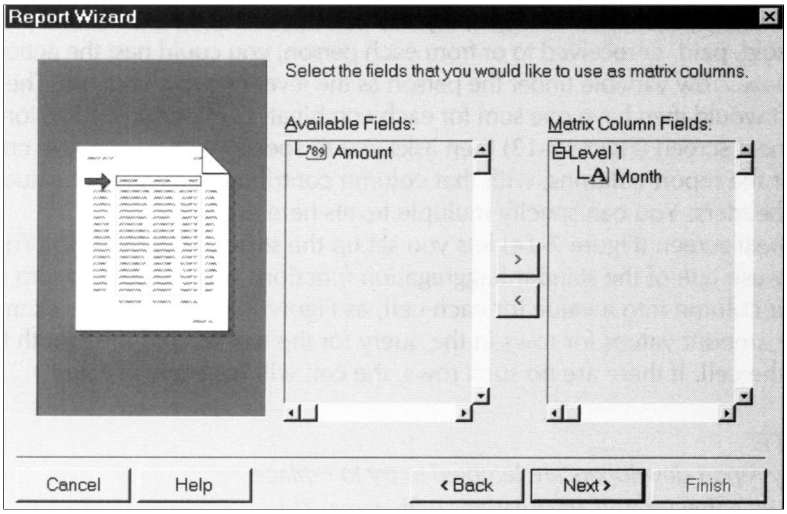


图7-29 矩阵报表的Column屏幕

很多报表开发者喜欢用“零”代替汇总值的空值(null)。这会导致一个令人遗憾的错误解释，尤其是当零是一个有效的汇总值时。如何区分“真正”的零与假零呢？在此例子中，“该月无费用支出”与“该月支出$0”实际上相同的，但不能总认为说两者是相同的。

统计基础所教的第一件事情就是：包括空值的行汇总是一个空值，并且不能把空值看成零值。只有在限制性非常强的环境，才能断定一个空值与零是相同的。应尽量避开这种易混淆的问题。可以单独保留空值，或者合适的话使用零值而不使用空值。例如，如果某些人没有得到Commission(佣金)，不是让Commission(佣金)为空，而是在数据库的该行中把它赋成零。避免这种问题的另一个原因是，Oracle Developer是相当难实现的。实现时，必须在重复单元框架中插入一个文本‘ 0 ’，然后设置填充该域，

China-pub.com 下载

以便在该单元有值时隐藏‘0’值。

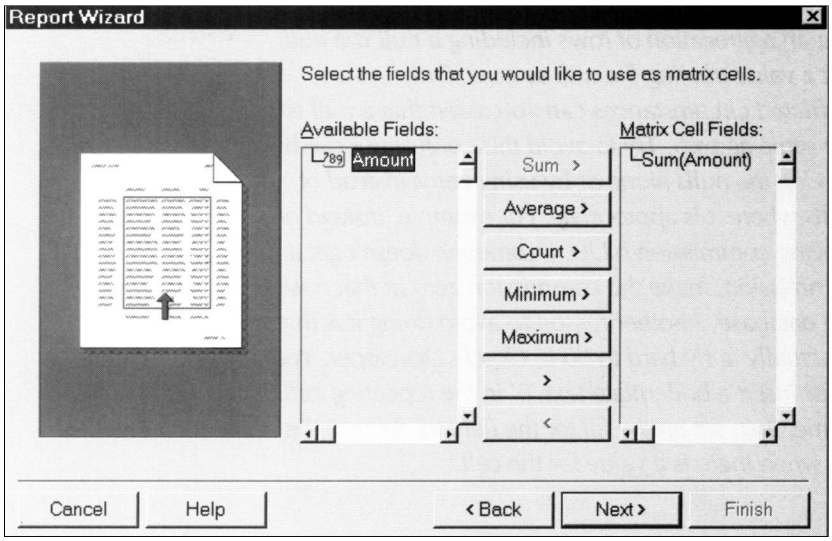


图7-30 矩阵报表的Cell屏幕

标准Wizard Totals屏幕允许指定求总计的列。设置总计显示，可以在矩阵的右边显示行的总计，在底部显示各列的总计，在右下角显示整个表的总计。这种情况下通常是使用某一单元列的和或使用几个列的和。

标准Wizard Label(标号)屏幕允许修改各列的标号，在报表中这些标号是非常特殊的，因为这些列及行标号都来自于数据。我喜欢整个删除列标号，因为它只是在数据标号下的简单重复。最后，Template屏幕允许为报表布局选择一个模板。图 7-31是在调整单元大小及标号之后报表显示结果。

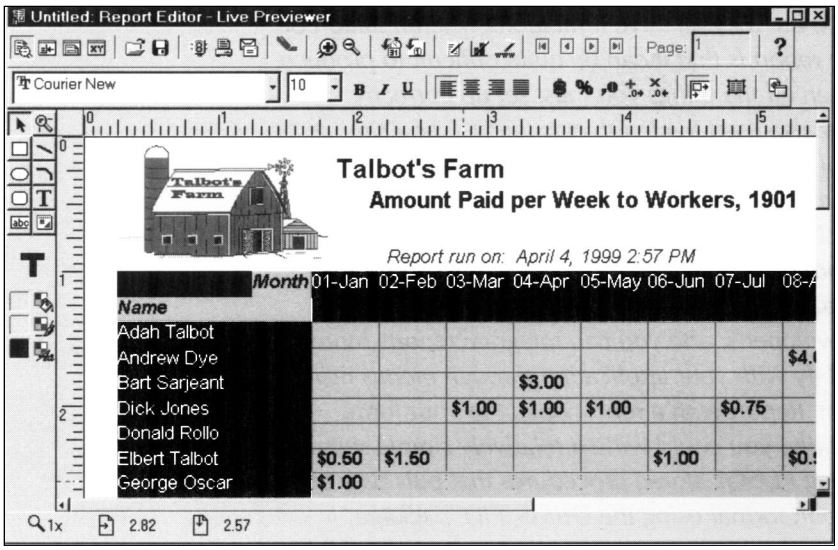


图7-31 Worker Paid矩阵报表

为了了解报表的实际内容，图 7-32列出了Report Wizard生成的数据模型。可以看到，依

China-pub.com
下载

赖于查询的有两个分组，分别对应着矩阵的行和列。在数据模型中的新内容是一个分组包围着两个分组。这种分组是一种交叉乘积组，它的功能是对分组的不同值产生交叉乘积。交叉乘积组的元素就是矩阵报表的各个单元。

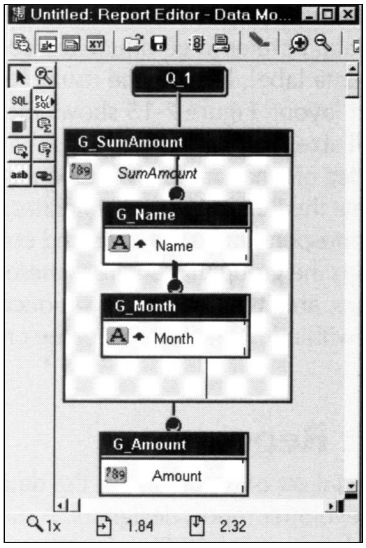


图7-32 Worker Paid矩阵报表的数据模型

7.3.5 数据文件导出报表

数据文件导出报表尽管不在正规报表模式集中，但在Oracle Developer的报表设计中是用得很多的。数据文件导出报表(data file export report)是一种输出普通文本的ASCII数据文件，通常用于输出到其他软件系统中。例如，一种目的是输出数据到电子表格上用于进一步分析，另一种目的输出报表到字处理器进行格式化。产生这种报表稍有难度，仅仅使用Report Wizard是不行的。

注意 有几种不同的方法可以完成这种功能，但都有局限性。报表的限制是产生所要求的内容十分困难，并且需要正确地设置。也可以使用Oracle OLE Object或类似的集成机制直接从数据库中取数据，但这只适用于支持这样机制的系统。用 SQL*PLUS也可以产生类似的ASCII报表，但它的报表能力与Oracle Developer相比是非常有限的，在表单中通过菜单用Run_Report显示一个报表对象，可以使报表与应用程序结合在一起。最后，也可以使用PL/SQL存储过程写一个相对简单的系统，该存储过程使用DBMS_FILE程序包以输出格式输出数据。

如下例子从Person表产生一组数据，列表显示表中所有人的姓名、年龄及住所。在新报表的Report Wizard中，在Style屏幕选择表格式报表，然后在 Data屏幕输入下面的SQL SELECT语句:

```
SELECT '"'||p.Name||'", '||
          p.Age||',"'||l.Lodging||'"' Output
  FROM Person p, Lodging l
 WHERE p.Lodging = REF(l)
 ORDER BY Name
```

这个SELECT产生一系列用逗号分隔的合并列，每个列的值用一对双引号括着。这种格式，叫做逗号分隔的数据输出格式，是很多软件系统使用的通用的方式。逗号定界各个值，引号保证输入数据的软件能正确处理空格、逗号及空值。“ Output ”别名在后面的Wizard屏幕中会用到。

注意 如果在列值中嵌入了双引号，必须用SQL函数把它们翻译成其他符号。

1) 在域及标号屏幕中，在报表中包括Output域并删除标号，保留它为空值。
2) 给Output域规定一个合适的数据行长度，如80或132。
3) 在模板屏幕选择不使用模板，如果已开发了模板，则使用数据输出文件模板。
4) 完成报表制作。
5) 在Object Navigator中选择报表对象并显示Property Palette。

China-pub.com 下载

6) 找到Charter Mode标题及Design in Character Units属性，并设为Yes。

7) 找到系统参数标题(System Parameters)，它在Object Navigator Data Model对象下面。部署该标题，会看到报表的所有系统参数。双击参数，会看到参数的 Property Palette。需要为有关系统参数设置一个初始值，以便给 Oracle Develoepr提供创建输出文件所需的信息，如表7-1所说明的。最后必须删除报表的边界。

8) 显示报表的Layout Editor，并单击Margin工具，会显示一条表示边界的黑线。

9) 单击该线，选择边界并显示改变大小的手柄。

10) 分别拖住四条边的中部手柄往边界拖。直到将要移出“纸”的边界止。例如，把上边界拖到Editor的顶部，左边界拖到左边线。

表7-1 Data File Export Report的系统参数设置

系统参数	初值样例	描 述
DESNAME	C:\DataFile.tx	输出文件名，包括路径
DESTYPE	File	“File”目标类型告诉Oracle开发器产生文件输出，而不是发送到打印机上
MODE	Character	告诉Oracle Developer产生ASCⅡ字符输出而不是标准的打印机输出
DESFORMAT	Delimited	告诉Oracle Developer以适当的格式输出以便接受定界域格式的程序输入数据，该域没有下拉列表，应确保输入正确的格式

11) 保存报表，单击Green-Light运行它。如果查看说明的输出文件，会发现如下的报表：

```
"Adah Talbot", 23,"Papa King"
"Andrew Dye", 29,"Rose Hill"
"Bart Sarjeant", 22,"Cranmer"
"Blacksmith", ,""
"Boole And Jones", ,""
"Dean Foreman", ,""
"Dick Jones", 18,"Rose Hill"
"Donald Rollo", 16,"Matts"
"Dr. Carlstrom", ,""
"Edward Johnson", ,""
"Edythe Gammiere", ,""
"Elbert Talbot", 43,"Weitbrocht"
"Feed Store", ,""
"Fred Fuller", ,""
"Gary Kentgen", ,""
"General Store", ,""
"George August", ,""
"George B. McCormick", ,""
"George Oscar", 41,"Rose Hill"
"Gerhardt Kentgen", 55,"Papa King"
"Harold Schole", ,""
"Helen Brandt", 15,""
"Henry Chase", ,""
"Isaiah James", ,""
"James Cole", ,""
"Janice Talbot", ,""
"Jed Hopkins", 33,"Matts"
"John Austin", ,""
```

China-pub.com
下载


```
"John Pearson", 27,"Rose Hill"
"Kay And Palmer Wallbom", ,"Rose Hill"
"Lily Carlstrom", ,""
"Livery", ,""
"Manner Jewelers", ,""
"Methodist Church", ,""
"Mill", ,""
"Morris Arnold", ,""
"Palmer Wallbom", ,""
"Pat Lavay", 21,"Rose Hill"
"Peter Lawson", 25,"Cranmer"
"Phone Company", ,""
"Post Office", ,""
"Quarry", ,""
"Richard Koch And Brothers", ,"Weitbrocht"
"Robert James", ,""
"Roland Brandt", 35,"Matts"
"Sam Dye", ,""
"School", ,""
"Underwood Bros", ,""
"Verna Hardware", ,""
"Victoria Lynn", 32,"Mullers"
"Wilfred Lowell", 67,""
"William Swing", 15,"Cranmer"
```

本章已经深入学习了如何生成几种不同类型高效报表的方法。当准备发布报表时，浏览一下第14章，该章描述了Oracle Developer Server及Report Server组件。不要忽略了在第4章学习过的产生用于因特网的 HTML及Adobe Acrobat格式报表的技术。第 8章进一步介绍Oracle Developer的图形组件方面的有关知识。

China-pub.com
下载

第8章 高 级 图 形

Oracle Developer的第三个主要组成部分 —— Graphics，通过使用生动的图形给应用程序增添趣味。多媒体是因特网出版物中游戏的名字 (原文如此 —— 译者注)，Graphics提供了方便地设计非常漂亮的应用程序的能力。通过 Graphics Runtime系统(图形运行时系统)，既可将这些图形作为独立的应用程序显示，也可将其作为表单或报表的内嵌图表。

注意 如果喜欢PL/SQL编程，并且准备迎接挑战，则可以使用Graphics中的编程系统创建所能想象出的任何图形显示。只要愿意去工作并设计出所需的几何和逻辑需求，就可以利用Graphics库中的数百个可用子程序去绘制和修改成所期望的任意形状。关于这种高级程序设计能力的详细说明参见Graphics的联机文档。

本章将介绍一些在应用程序中创建图形的基础知识， 8.1重点介绍使用图形的方法，其他几节详细介绍如何使用Graphics Builder创建所需的图表。

8.1 有效使用图形

人们都喜欢看有大量图形和可视对象的应用程序。深入了解图形效果及其表示数值的结构，将有助于创建需重使用的图形对象。

应用程序中的图形是表达信息的一部分形式，而并非仅仅是装饰物。一个图形就像一个单词，只有表达清楚才能有效地传递它的信息。大部分人在他们的职业生涯中都接受过最基本的语言表达和写作能力方面的培训，但很少有人接受图形或图形结构方面的培训。为多学些这方面的知识，本节提出一些建议和参考。然后本章将重点介绍用 Oracle Developer的Graphics能够创建的图形。

首先要说，图片能干什么？一张图片能代替千言万语，对吗？很遗憾，但愿它能传递千言万语。如果所涉及的是要表达一个简单的事实，采用图片显示就有点大材小用了，而且也许还会妨碍正确信息的表达。在应用程序中使用图片只是表示那些真实的信息，例如，不动产查询系统中的一个房子或公寓的图片、人员资料数据库中的个人图片、销售应用程序中的产品图片。

其次，位图标志和图标能干什么？位图的典型用处是显示公司的标志，如第 4章的Talbot报表。要和公司交往联系的人，只需通过应用程序，就可形象地了解到与他合作的公司是什么样子，这是个好办法。然而，只在机构内部使用的应用程序，这些商标可能就是多余的。例如，在备忘录上，公司微标表示“法定的”的表征特性，这些图像与其说是必需的，不如说是为了装饰用。

图标的另一个主要用途是代表某些初始化动作的按钮，显示某个窗口等等。例如，在工具栏中使用图标已经相当普及。要在应用程序中创建大量的控制位图，必须确认这些位图不影响应用程序的信息表示。图标要尽量小而且慎重使用，或者将图标单独放在自己的窗口中(工具条是一个存放图标的好地方)。要想使图标发挥作用，在设计和测试时要花费一些时间。

China-pub.com
下载

现在来深入讨论复杂难懂的图形显示方面的问题。用户首先要决定的是：哪种图形适合自己的应用程序。要完成这一步，必须先确定希望通过图形传达一些什么信息。正如在一段文字中要清楚地表达出自己的论点一样，在图形中也要尽可能清楚地表示出主要内容。如果搞不清楚主要内容是什么，就不要使用图形。另外，还应尽量将主要内容概括成图表的标题或题目，告诉读者图表的含义。

一旦设计好主要内容后，下一步必须确定要表达多少信息成分。曾经将一件事与另一件事比较过吗？根据日期比较过两件事吗？如有三组以上的信息，可能需要一个以上的图表来显示这些信息。

接下来，对于每一组信息，必须确定它们是定性的还是定量的。定性数据需要进行归类，如Talbot货物供应商的姓名。定量数据要用数字表示，如日期或数量。有的定性数据有隐含的顺序，如按字母排序的名字、年龄分组、经济水平层次等。

现在可以根据以上信息决定使用哪种图形了。例如，要显示一组定性数据和一组定量数据组成的二维图形，应当构造一个饼图、条形图或柱状图。用图形中的饼块及条形直接有效地表示分类。

第4章显示过的销售饼图 (图8-1)表示了这种二维的定性 /定量值图表。定性部分是人，定量部分是此人的销售数量总和。此图的主要内容是比较Talbot农场最大顾客占总销售量的比例，因此图的标题是“ Talbot顾客销售比例图 ”。

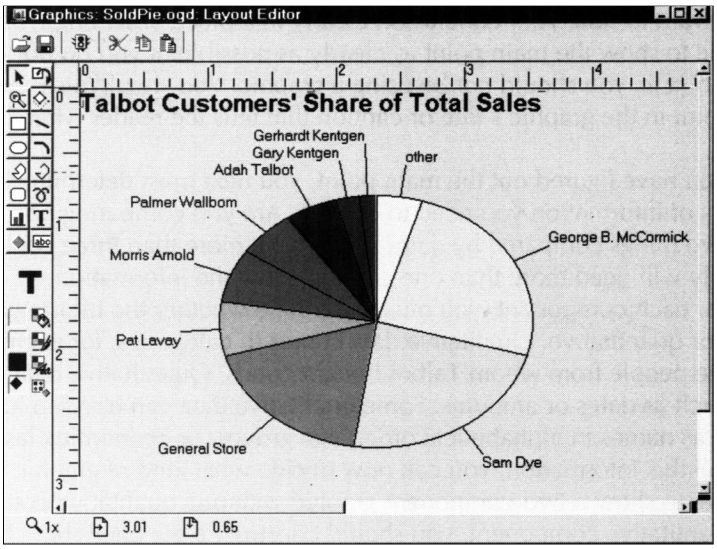


图8-1 表示Talbot顾客份额的销售饼图

如有多于一组的定量数据，就不能使用饼图表示，必须使用双条图表或柱形图，如果表示数量增加，可堆叠各条形柱，如果要比较它们，也可以一个挨一个放置。例如，如果想比较不同类型Ledger活动(Sold、Received、Paid、Bought)的现金流通情况，可使用图 8-2的条形图。

如果比较不包含类别数据的两类数字的相对关系，可以使用折线图或分布图来表示数据的变化趋势，或用一个表来表示准确数值，如图 8-3用三种样式的折线图表显示 Talbot农场某段时间的总销售量。对于总量的变化趋势使用折线图的阴影区来强调总量的大小。

China-pub.com 下载

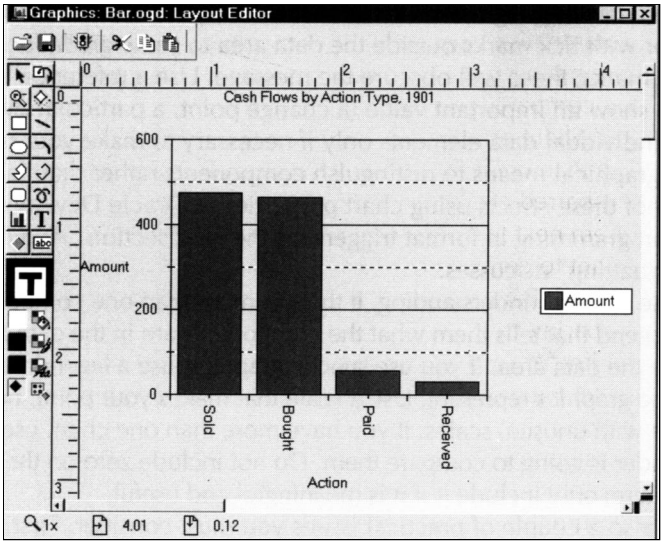


图8-2 Talbot农场现金流量类型

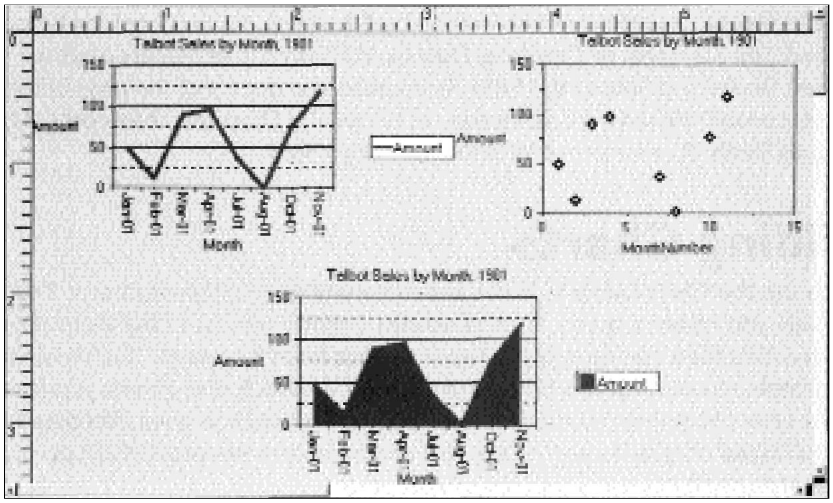


图8-3 Talbot农场销售趋势

尽管图表可加许多细节，但原则是应尽量简单。重点是数据，所以要确保细节不要影响数据。若有太多的标记，或需要一个复杂的图例，或有太多的组成部分，应先简化图表，否则便不能表达主要内容。将数据元素变成图表中最突出的元素。用合适的刻度线包围数据区，或用数据区外的刻度标志显示数据级别。不要使用太多的刻度标志，以免信息不清楚，图中用一参照线显示重要数据(一个变化数据、一个特定事件日期等等)。如果需要强调，可标记个别的数据元素。可能的情况下，尽量不用标记而用图形区分各个组成，使用 Oracle Developer 中图表属性可完成以上大部分功能，余下的功能需要用触发器编程完成，参见后面“为设置和格式化增加触发器”。

如果有多于一个的成分，为了帮助读者理解，可以用标题和图例说明数据区中的表示。图例要放在数据区外。如使用专门的图表，要使用图例说明图表表示的含义。应使用比例画数据点，但要避免用不常用的比例方式愚弄读者。如果有多于一个的图表，应该采用相同的

China-pub.com
下载

比例，以便读者可以比较它们，比例线上不要包含 0，除非0有用处和有意义。

还有两个必须考虑的实际问题。首先，应该核对或测试图表显示，确保达到目的而没有明显的错误。其次，要考虑如有人要打印此图形的黑白颜色的拷贝，可能会发生什么情况。一个在监视器上看起来很漂亮的图表，当从打印机打印出来时却变成一团黑色污点，这可能会使某个职员变成该图表的受害者。

若想进一步学习如何生成有效的图形显示，参考 Edward R.Tufte的《The Visual Display of Quantitative Informatin》(Graphics Press, 1983)一书，以及 William S. Cleveland的《The Elements of Graphing Data》(Wadsworth Advanced Books and Software, 1985)一书。要进一步深层研究这领域，参考 Jacques Bertin的《Semiology of Graphics: Diagrams, Networks, Maps》(University of Wisconsin Press,1983)一书。

8.2 创建图表

使用Chart Genie可创建许多不同的图表。这个向导首先创建一个查询，生成一个所要显示的图表的快速版本。然后可修改主要内容区的结果。第 4章已介绍了一个这一过程的简单例子，这个例子同时也介绍了如何将图表插入表单和报表中。这一节将详细介绍可以建立的查询和图表类型，以及可为每个类型图表设置的属性。

Oracle Developer显示模板包括在图形(Graphics)组件中定义的查询和布局。一个图表是带有显示区的一个特殊对象，并且显示区可以有一个或多个图表。另外，还可以有多个查询，不同图表使用不同的查询或相同的查询，或同一查询的过滤子集。例如，可有一系列的饼图，每个饼图说明数据的一个不同子集。或者使用一个饼图和一个折线图，说明一个变量与其他变量之间的关系。在 Chart Genie中分别定义每个图表，然后用 Layout Editor排列图表，在整个显示区，为区别每个图表的属性，可为图表添加标题和图例。插入图形时，是在显示区插入的整个图形显示，而非图表本身。

要使用数据的查询结构决定要建立的图表，同样，图表结构取决于数据结构。创建图表的大部分工作是获得正确的查询结构。这个工作要多动些脑子，集中精力于定性 /定量的区分，粒度实验和数据分组。

下节讨论创建查询，然后讨论创建图表的特殊类型。

8.2.1 为图表获取数据

第4章已介绍了如何使用 Chart Genie和它的New Query屏幕为图表构造一个基本查询和数据模型，本节对构造过程做更为详细的描述，对第 4章演示的用Chart Genie建立查询的过程，给出一些其他选择方法。最后一小节介绍用单个查询关联多个图表进行数据过滤的方法。

1. 选择数据源

创建图表时，Oracle Developer并不限制在Chart Genie中键入一条SELECT语句，有多种不同的方法可为图表获取数据。

注意 高级用户可以在Object Navigator中，通过查询对象的Property Palette直接创建查询。也可使用PL/SQL内部子程序，如OG_Make_Query和OG_Execute_Query创建查询。还可通过所建查询的Data标签页手工输入数据，尽管这是个笨方法。

1) 使用外部查询 和报表一样，图表也可使用外部查询获得数据，在第10章的“外部查询

China-pub.com 下载

和Query Builder ”中详细介绍Oracle Developer中报表组件的外部查询模块。在Graphics中，外部查询不是一个模块，它是一个可以通过Chart Genie的Query页面中New Query对话框访问的文件。

有两种使用外部文件的方法：引入查询和引用外部文件。这两种方法都需要放在 ASCII文件中的一条单个的Select语句。不要像在一个SQL*plus脚本中一样使用分号结束语句及包含程序选项。

复制语句文本可将SQL语句引入查询中，单击SQL Statement文本框右边的Import SQL按钮，如图8-4所示。

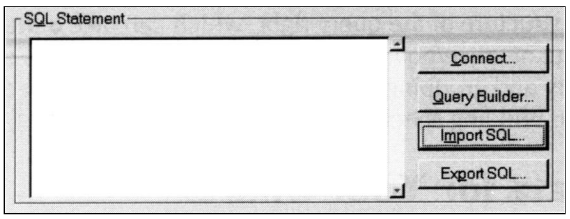


图8-4 SQL Statement文本框

这个按钮打开一个标准的 Open对话框，允许选择文件，然后复制文件内容到 SQL Statement文本框，然后就可以如同自己输入的SQL语句一样处理。

注意 这个方法有限制，用户看不到SQL Statement文本框中的SQL语句。单击Execute进入Data页可看到结果数据。单击Export SQL按钮也可将自己输入的SQL语句输出到一个外部文件中，这是一种创建能被其他图表和报表重用的语句的简单方法。

在外部文件中使用 SQL语句的最好方法是使用 Type下拉列表框选择查询类型为 External SQL File(外部SQL文件)。这个选择激活File域和Browse按钮，如图8-5所示，Browse按钮通过标准Open对话框选择文件。

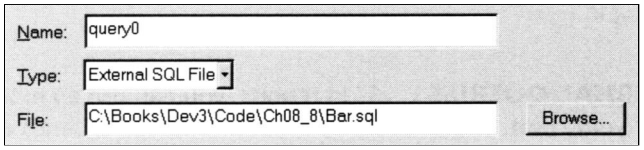


图8-5 Browse按钮通过标准Open对话框选择文件

此时不用复制SQL语句。当需要执行SQL时，Chart Genie引用此文件。这意味着，如果改变文件内容，图表也相应改变，这将提高查询和图表的重用能力和可维护性。

2) 从文件中获取数据 另外还可从几种传输数据格式文件中获取数据：

- SYLK：一种旧的Microsoft Excel/Multiplan输出文件格式。
- WKS：一种旧的Lotus1-2-3输出文件格式，也被其他传输协议和数据库程序使用。
- PRN：未知结构的通用打印机输出文件 (本人不推荐这种选择，因为除了是文本格式外，它不属于任何文档)。

从下拉框中选择合适的格式，如图 8-6所示，然后使用激活的Browse按钮打开标准Open对话框指定文件，数据将显示在 Data标签页上。

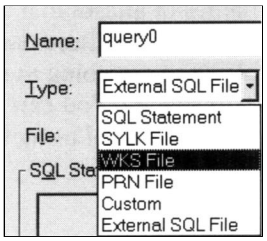


图8-6 从下拉框中选择合适的格式

China-pub.com
下载

注意 利用查询中可使用的有限的数据格式向Oracle表中引入数据，要比用非SQL外部文件格式好。可以使用这种将文件格式转换成 Oracle表的产品，比如Data Junction by Tools and Techniques等，自动完成这个过程。

3) 从调用模块中获取数据 当其他模块嵌入到图表时，常常是要用图表查询与调用模块相同的数据。例如，一个表单或报表以表格形式显示数据，其嵌入图表以图形形式显示同样的数据。在表单中可为 OG.Open调用建立一个参数表。第 10章的“为报表和图表传递参数”节将详细介绍使用数据参数将 Forms的记录组作为查询传递给图表使用。

要想有效地使用参数，必须协调地显示模块与调用模块：

- 在Chart Genie的Chart Properties对话框或Chart Properties表中，找到Data标签页，检验标有“Update chart on query execution”复选框。如图8-7所示，这个复选框告诉显示模块，当调用模块传递控制给图表时，修改显示结果。 Oracle Developer中此复选框的缺省状态为选中。

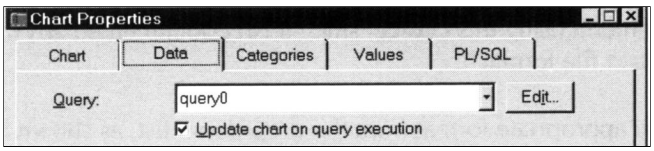


图8-7 选择Update charten query execution复选框

- 通过数据参数传递的记录组中的列与查询中列的名字或别名必须相同，显示模块必须匹配这些名字。显示模块查询中的别名，如果要通过内部控制图表来控制，则该别名要易于识别。

2. 过滤数据

通常，使用SQL语句的WHERE子句来过滤显示在图表中的数据。如第10章中讨论的那样，可通过词法引用参数对 WHERE子句参数化。然而，在特定情况下，可能需要使用 PL/SQL的过滤功能来过滤记录：

- 使用的数据是非SQL文件格式，或数据是手工键入到查询的。
- 使用调用模块的数据，但希望用图表控制记录集的内容，而不用调用模块控制。
- 显示窗口中的多个图表使用同一查询，但需使用不同的数据子集 (需要对数据库中相同数据进行多次无效查询)。

进入Chart Properties对话框的Data标签页，找到Filter域，如图8-8所示。

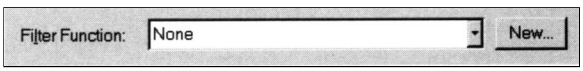


图8-8 进入图表属性对话框的Data标签页

从已有的过滤函数下拉框中选择，也可单击 New按钮创建一新的过滤函数。如果是创建新的函数，则Graphics显示一个带过滤函数模板的PL/SQL Program Unit Editor：

```
-- Query Filter Functions. Called for each row of a query
-- that is associated with the chart object
-- ARGUMENTS:
--   CHARTOBJ  The current chart object.
--   QUERY     The query associated with the chart object
```

China-pub.com 下载

```
-- RETURN:
--    TRUE    keep the row
--    FALSE   remove the row
FUNCTION OGQUERYFILTER0(chartobj IN og_object,
                        query IN og_query)
                        RETURN BOOLEAN IS
BEGIN

END;
```

可在BEGIN与END语句间添加代码，根据查询数据列的值返回 TRUE或FALSE。执行查询时，Graphics对每个记录执行该函数，过滤掉函数值返回 FALSE的记录。

8.2.2 创建折线图

一个折线图显示两组以上的数值数据 (图8-9)，用线条连接数据点。这种图表的不同子类型使用不同的方法连接数据点。例如，一个散点图只显示数据点，一个折线图用曲线连接数据点，一个面积图显示线条下面连续的区域。 X轴代表一个数量变化， Y轴代表进行比较的一个或多个数量变化。图 8-9以折线图、散点图和面积图的形式，显示 1901年月销售量与时间的关系。

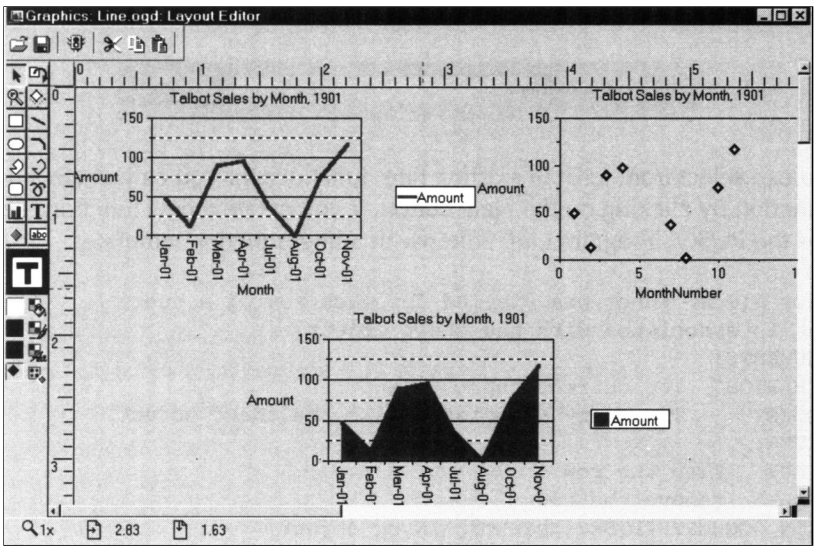


图8-9 折线图、散点图和面积图

创建折线图的关键是确定有正确的坐标间隔的两组定量数据，如图 8-9所示， X轴刻度是月，Ledger表中的原始数据是每一天的销售量。要统计出每月数据并正确排序，需使用下面的查询：

```
SELECT To_Char(ActionDate, 'Mon-YY') "Month",
       To_Number(To_Char(CctionDate, 'MM')) "MonthNumber",
       Sum(Amount) "Amount"
  FROM Ledger
 WHERE To_Char(ActionDate, 'YYYY') = '1901' AND
       Action = 'Sold'
 GROUP BY To_Char(ActionDate, 'Mon-YY'),
          To_Number(To_Char(ActionDate, 'MM'))
```

China-pub.com
下载


```
ORDER BY 2
```

在选择列表中的MonthNumber列，虽然未出现在图表中，但却是按照它进行排序的。因为若按Month列排序，则只会以字母顺序排序，而不能按月份值排序。 WHERE子句只将1901年的销售数据取出来， Group BY子句包含Month和MonthNumber两列，以便选择列表能够包含这两列。带分组子句的SELECT语句的选择列表中，只能包含求和函数表达式或GROU PBY子句中的元素。通过Chart Genie的属性窗口可设置图表的分类和列值。

既然已经掌握了基础，那么如何修改图表的格式和布局呢？

8.2.3 修改图表

Chart Properties对话框控制每个图表的基本属性。要打开这个对话框，首先通过单击图表来选择图表，然后用 Chart|Properties菜单项，或者通过右击弹出菜单的 Properties打开Chart Properties对话框。在对话框中选择图表的缺省类型和子类型，通过前面见过的标签页建立查询与图表间的关联，这些标签页是用其建立条形图或饼图的 Chart Genie序列的一部分。

1. 修改图表框和坐标轴

要了解下一层的图表格式，选择 Chart或弹出菜单上Frame(图框)或Axes(坐标轴)菜单项。当选择图表及Frame后会出现Frame Properties对话框，可以用于控制图表框的外观。图 8-10的Frame标签页允许修改外框的基本属性：

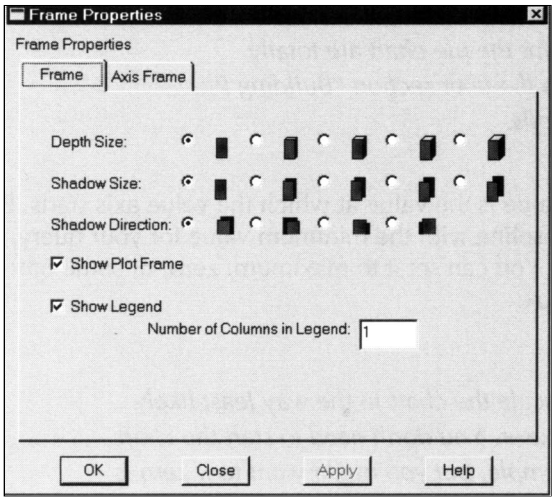


图8-10 允许修改外框基本属性的Frame标签页

其中，三组单选按钮控制图表框的三维外观和阴影特征，如，可将外框设置为将图表数据放在中央的盒状。 Show Plot Frame复选框用于隐藏或者显示外框边界线。 Show Legend(显示图例)复选框决定隐藏或显示图例盒。

图8-11的Axis Frame(坐标轴框)标签页用于修改坐标轴属性。

注意 饼图的标签页与此完全不同，详细内容参见后面的“创建饼图”一节。

基准值是坐标轴的起始值。其缺省值是查询的最小值，通常也是正确设置。使用此对话框还可将基准值设为最大值、零或其他固定值。

注意 记住，以这种方式按照比例缩小图表时有可能使用户搞不清楚。图表不必以零值

China-pub.com 下载

开始，但是若零是个有效基准值，必须以零开始。例如，在一个显示窗口中有几个图，可能想将它们的基线值设成同一值，比如零，以便能够准确比较。

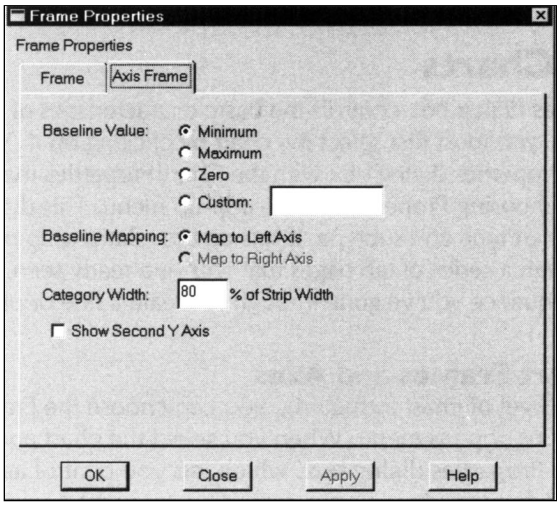


图8-11 Axis Frame标签页

若单击Show Second Y Axis(显示第二个Y坐标轴)复选框，图表在第一个Y轴的另一边显示第二个刻度线。Baseline Mapping(基准映射)单选按钮控制每组值到刻度尺的映射。例如，可以使用这个属性显示两条不同刻度的线。可以用两种完全不同的刻度相对于时间来比较两个变量的比率而不是绝对的值。Category Width(种类宽度)设置控制一个种类坐标轴的宽度。

当单击Axes菜单项时，会显示Axis Properties(坐标轴属性)对话框，对话框中有Axis标签页及Continuous Axis(连续坐标轴)或Discrete Axis(离散坐标轴)标签页，这取决于在下拉列表框中选择的坐标轴类型。如图8-12所示，Axis标签页控制坐标轴的标签和刻度的设置。

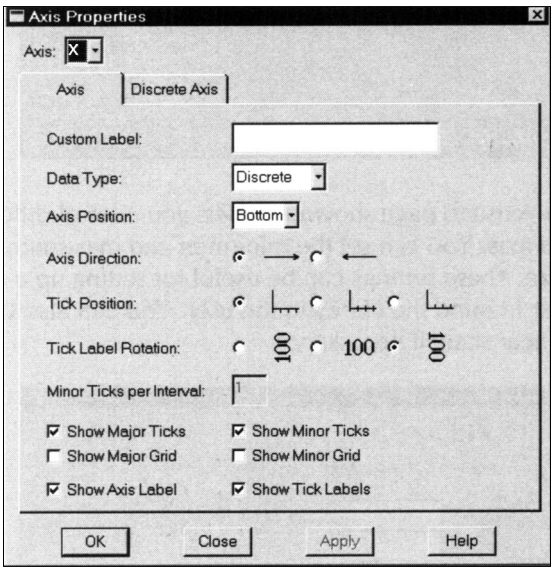


图8-12 Axis标签页控制坐标轴的标签和刻度的设置

在Custom Label(定制标签)域中输入自己的标签可以取代自动设置的标签。使用 Data

China-pub.com
下载

Type(数据类型)下拉框指定坐标轴是连续、分离还是日期型。设定结果必须与 Chart Properties 对话框中给坐标轴指定列的数据类型相符合。通过下拉列表框可以控制坐标轴的位置 (底部、顶部、左边或右边)。通过单选按钮可以控制坐标轴的方向 —— 是从左到右，还是从右到左。刻度和标签的各种设置为用户提供了更高程度上控制坐标轴和标签外观的方法。

在坐标轴上显示定性数据时，类别标记的个数可用如图 8-13的Discrete Axis(离散坐标轴)标签页控制，通常情况下最好设为 auto(自动)。

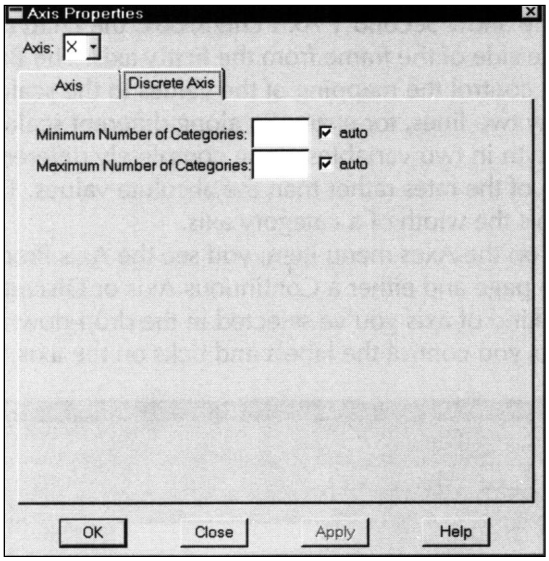


图8-13 Discrete Axis标签页控制设为auto

图8-14中的连续坐标轴(Continuous Axis)标签页控制坐标轴上不同数值特性，可设置坐标轴的最小值和最大值以及刻度间距。这些设置对于建立一个可比较的比例是有用的。但一定要记住，对用户来说必须是清楚的。若必要的话，还可使用对数比例而不用线性比例。

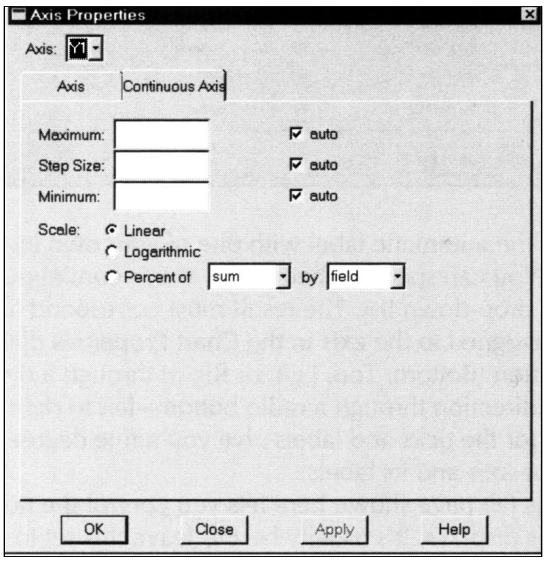


图8-14 设置连续坐标轴的值

China-pub.com 下载

既然已经学习了修改图表的所有方法，那么如何修改图表内容，即修改单个的图形对象呢？

2. 修改图表对象属性

要修改对象的格式属性，必须先选择对象。要选择对象，首先要选择图表，然后单击对象，如线条、符号、饼块或其他。如只单击图表对象一次，只是选择的图表本身，再次单击直到看到所有图表组件带有选择句柄为止。现在可使用 Format(格式)菜单和工具板中的格式化工具(线条、填充和符号)对图表的组件格式化，如颜色、线宽、斜角、箭头，以及字体设置等等。

通过域模板还可修改域的某些属性。可选择图表或域，然后单击弹出式菜单上的 Field Template(域模板)菜单项或Chart(图表)菜单。根据图表的类型，显示一个带标签页的对话框，通过它可以修改格式化域的模板。第 10章将详细介绍使用和重用域模板对多图表对象进行格式化。

8.2.4 生成条形图

一个条形图显示一个定性成分及一个或多个值成分 (图8-15)。条形的大小反映数量值。显示多个值成分的变化情况，可以在条形上作标记，或一个挨着一个地放置条形。图 8-15显示1901年现金流通类型与总数量关系的条形图。

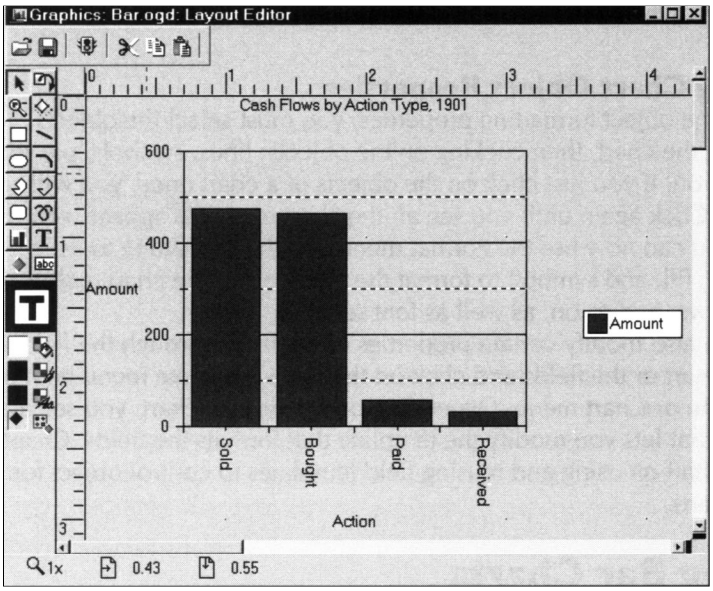


图8-15 条形图

创建较好条形图的技巧在于适当划分不同的定性类。图 8-15的条形图只需一个相对简单的查询：

```
SELECT Action "Action", Sum(Amount) "Amount"
  FROM Ledger
 WHERE To_Char(ActionDate, 'YYYY') = '1901'
 GROUP BY Action
 ORDER BY 2 DESC
```

China-pub.com
下载

GROUP BY子句设置条形图的分类维数， ORDER BY以降序方式对数量值进行排序，WHERE子句提取1901年的数据。

一个柱状图(Histogram)是一个横着显示的条形图，条形沿X轴从左到右伸展，而不像条形图沿Y轴从下向上伸展。通常，柱状图描述两个数据成分并显示一种概率或分布频率。图 8-16显示现金销售相对时间的频率，每个条形代表 1901年的一个星期，并且数据按下降的频率排列。这个柱状图显示现金销售的单向频率分布，表明这一年销售数据按星期的分布情况。如果按照销售数据的随机变化情况正常分布，柱状图看起来更像一个半自动的曲线。

相对数量的变化，柱状图显示一种频率，需要构造与条形图类似的查询：

```
SELECT To_Number(To_Char(ActionDate, 'IW')) "Week",
       Count(*) "Count"
  FROM Ledger
 WHERE To_Char(ActionDate, 'YYYY') = '1901' AND
       Action = 'Sold'
 GROUP BY To_Number(To_Char(ActionDate, 'IW'))
 ORDER BY 2 DESC
```

GROUP BY子句使用ISO星期编号，按照星期统计数据，星期编号成为数量值。频率成分来自于SELECT列表的Count(*)。ORDERBY按降序排序频率(见图8-16)。

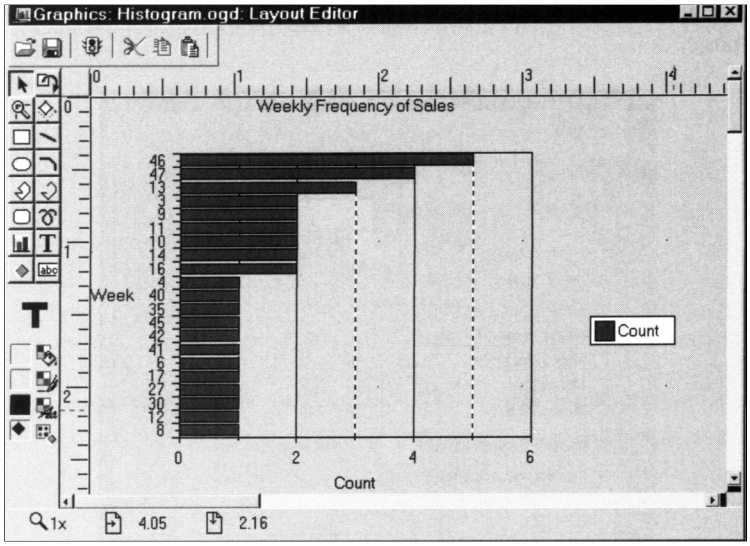


图8-16 相对时间的现金流量柱状图

8.2.5 生成饼图

一个饼图显示一个定性成分和一个数据成分，饼块的大小反映数量值，饼块表示这个值占总值的比率。有变化的图形反映不同显示风格，如带阴影的饼图及三维饼图 (见图8-17)。

利用饼图专门的 Frame Properties(框架属性)对话框可修改饼图的各种选项，其中 Pie Frame(饼图框架)标签页设置专门的饼图选项(见图8-18)。

在第4章已经看到“其他”饼块的创建方法，余下的选项则控制饼块的特定属性及饼块之间的关系。Pie Frame标签页中的选项控制标签的内容及位置。

和8.2.3节“修改图表”所描述的一样，通过标准的Frame标签页可控制饼图的深度和阴影，

China-pub.com 下载

Show Legend(显示图例)复选框对饼图无效。

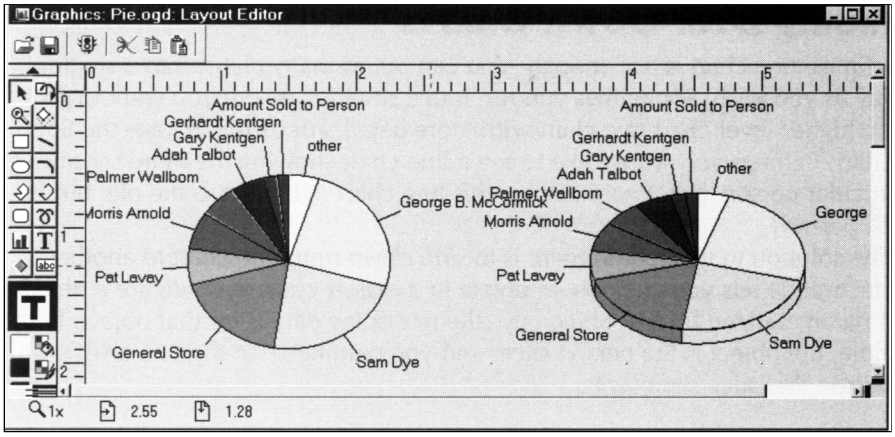


图8-17 饼图及三维饼图

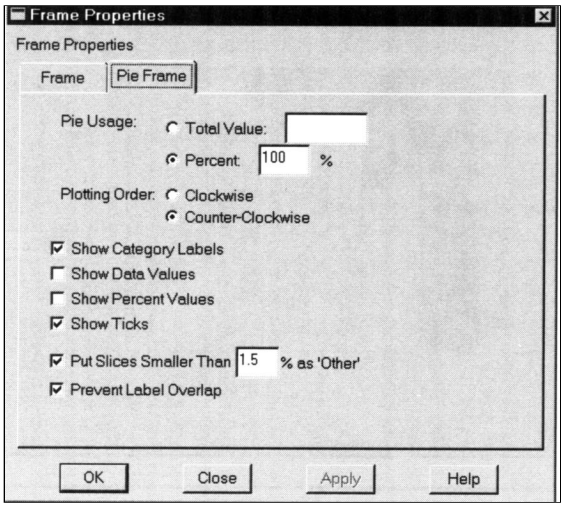


图8-18 Pie Frame标签页

8.2.6 建立下钻图表

有时，只有一个图表是不够的，可以在一个显示窗口放入多个图表。有时会遇到希望从一个上层图表进入一个细节图表的情况。就 Sold饼图为例，可能希望看到显示某人的销售趋势的折线图。可以将这个图显示在饼图的右边，但它是哪个人的呢？

要解决这种要求的方法是从一个图表中下拉出另一个图表。这种技术允许在主图表中单击一个对象，然后对应该对象打开下一个带参数的细节图表，通常是显示该对象的详细情况。示例中的饼块对象是人员，可将饼块代表的人员作为折线图的参数。

在Oracle Developer中创建一个下拉关系非常容易。无须任何 Graphics编程，只需两个图表以及对主图表进行一些属性设置。这种关系的关键是修改细节图表的查询，使它包含一个宿主变量引用基于主图表的值。例如，当单击某个饼块时， Graphics Runtime系统设置参数，执行查询，重新显示细节图表。

China-pub.com
下载

在Sold饼图旁增加一个折线图，如图8-19所示。

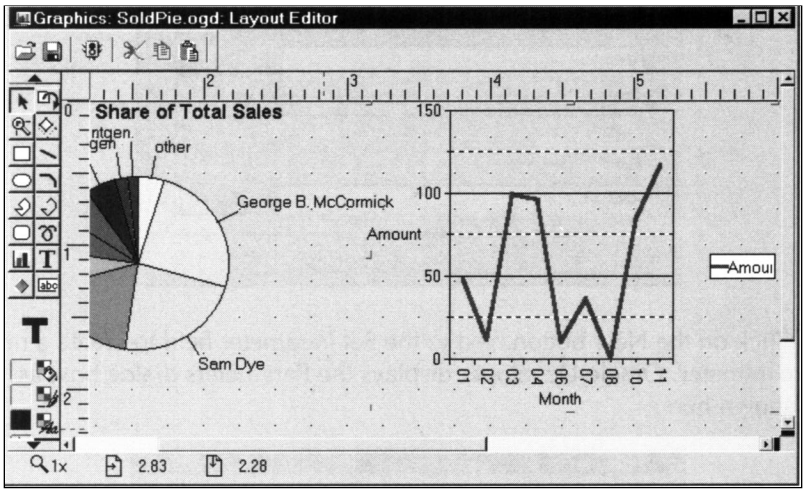


图8-19 Sold饼图旁增加一个折线图

这个图表是一个标准的折线图，通过 Chart Properties(图表属性)对话框将X轴上标签方向从水平变为垂直，除此之外没做其他修改。

产生此图的查询包含下面的SQL语句：

```
SELECT To_Char(ActionDate, 'MM') "Month", Sum(Amount) "Amount"
  FROM Ledger
 WHERE To_Char(ActionDate, 'YYYY') = '1901' and
       Action IN('Sold', 'Received')
 GROUP BY  To_Char(ActionDate, 'MM')
 ORDER BY 1
```

在这个图中查询为1901年每月销售量总和生成一个数据点，这个点并不与饼图上对应的人相一致。

1) 在Layout Editor(Layout Editor)中打开饼图，单击选中的饼图，然后单击所有饼块选择饼块对象，所有饼块周围出现选择柄。

2) 右击鼠标弹出菜单，选择 Properties(属性)菜单项，或从主菜单选择 Tools|Properties(工具 | 属性)菜单项。

3) 选择Drill-down标签页(见图8-20)。

4) 单击Set Parameter(参数设置)域旁的New(新建)按钮，创建一个新的参数。Oracle Developer显示Parameters(参数)对话框，如图8-21所示。

5) 改变参数名使其易读，如 PERSON，提供一个初始值如 George B.McCormick，他是最大的购买者。在这个例子中，数据类型已经是正确的 (CHAR)，如果有必要也可将其改为NUMBER或DATE。

6) 单击OK按钮，Oracle Developer在显

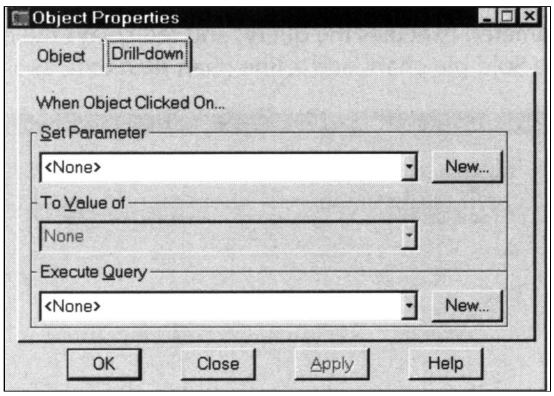


图8-20 Drill-down标签页

China-pub.com 下载

示模块中创建参数。

7) 回到Drill-down标签页，从To Value Of下拉列表框显示的列中选择Person。

8) 从Execute Query下拉列表框中，选择折线图的查询 (QUERY0)作为执行下拉图表的查询，设置后的标签页显示如图8-22所示。

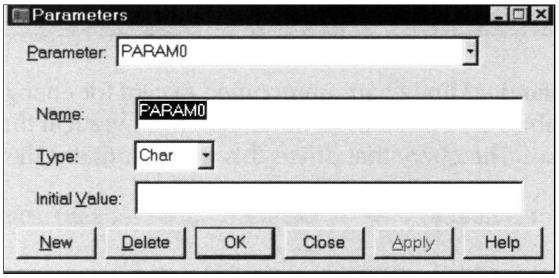


图8-21 Parameters对话框

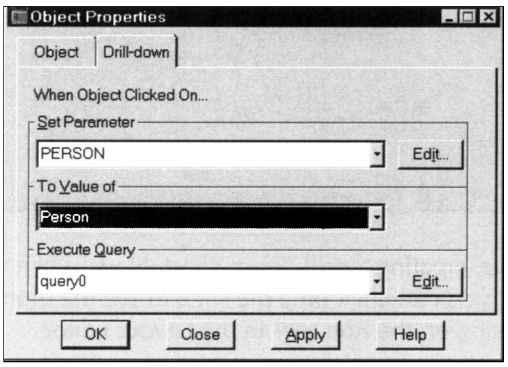


图8-22 设置后的Drill-down标签页

Execute Query域旁的New(新建)按钮变成了Edit(编辑)。

9) 单击Edit按钮编辑查询，增加一个到Person表的连接，得到Person Name列，然后，将这一列与:Person参数值比较：

```
SELECT To_Char(ActionDate, 'MM') "Month",
       SUM(Amount) "Amount"
  FROM Ledger l, Person p
 WHERE l.Person = REF(p) AND
       To_Char(ActionDate, 'YYYY') = '1901' AND
       Action IN ('Sold', 'Received') AND
       p.Name = :Person
 GROUP BY To_Char(ActionDate, 'MM')
 ORDER BY 1
```

10) 执行查询。查询语句将得到赋给参数的缺省值，这个例子中， George McCormick购买的最多，但主要是发生在一年的两个月内，刷新后的图形如图 8-23所示。

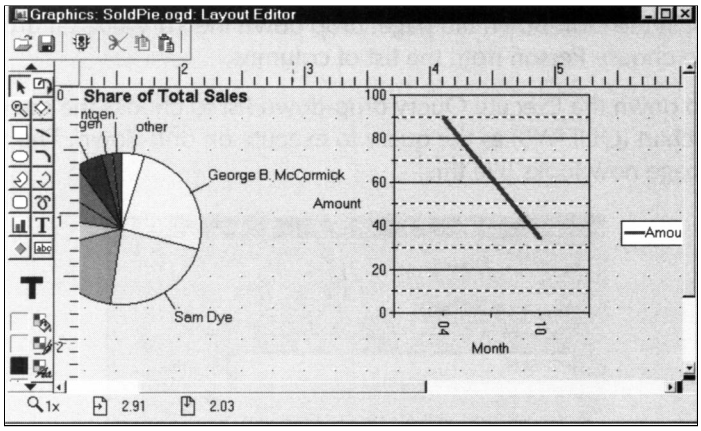


图8-23 刷新后的图形

以上是创建一个下拉图表的步骤。当在 Graphics Runtime中运行图表时,单击饼块可以查看

China-pub.com
下载

任何一个人的月销售趋势。请单击Layout Editor中的RUN工具试一试。

8.3 显示布局设置

Graphics Layout Editor有很多可用的工具，用于增加或修改图表的基本布局，下面介绍可利用编辑器完成的内容。

8.3.1 绘图、图像及声音

可以添加背景图形、位图映像和声音渲染，或者为其他目的处理基本图表，或者根本不用任何数据模型而使用这些元素构造一个完整的图形。

注意 有两个应掌握的绘画技巧。第一，双击工具面板中的工具可设置此工具绘制多个对象，单击意味着绘制单个对象，以后Editor会重新设置工具。第二，不论何时绘制一个图形，按住Shift并单击可绘制一个更规则的图形，如一个正方形而不是矩形，或生成一个圆形而不是椭圆。

单击图形工具可增加一个图形，如 Rectangle(矩形)、Line(线条)、Ellipse(椭圆)、Arc(圆弧)、Polygon(多边形)、Polyline(多边线)、Rounded Rectangle(圆角矩形)或Freehand(徒手画)。多边线由一系列线组成,边界外有一缺口，而不是一个完整的多边形。徒手画可在任意方向绘制像素点序列。一旦单击相应工具就可绘制相应图形。要修改图形的特征，选择并单击它。然后用合适的工具或菜单项修改它。工具板上有如图8-24所示的工具：

- 旋转 (Rotate)：拖动图形的尺寸柄可以旋转图形，按住 SHIFT 后拖动一次旋转45度角。
- 修改图形(Modify Shape)：修改多边形、多边线或徒手画图形。
- 填充颜色(Fill Color)：改变图形的填充色或使填充透明 (No Fill)。尽管本书建议如果用户接口有移植性要求不要使用任何填充图案，但也可设置填充图案。
- 线条颜色(Line Color)：改变图形边界线的颜色或使边界线透明(No Line)。

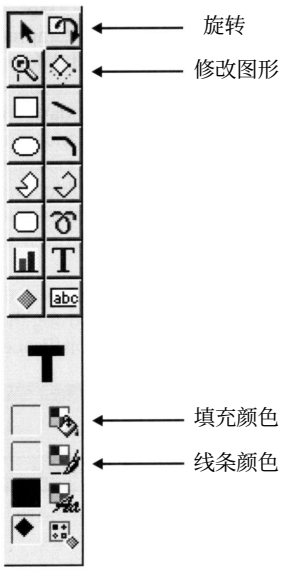


图8-24 工具板上的工具

能影响图形的一大组菜单项都在Format(格式)菜单上，如，可调整线宽、使用几种虚线模式之一、增加斜面、增加箭头、使符号变大或变小。通过选择 Drawing Options(绘图选项)菜单项，可访问一系列的对话框，实现对图形细节的完全控制。同时也有圆弧和圆角矩形的特定绘图选项，这两种图都有各自的特点。

8.3.2 为设置和格式化增加触发器

第5章详细介绍了Graphics Runtime的执行顺序，包括在不同点触发的各种触发器。本节介绍这些触发器的一些使用方法，着重介绍一些标准图表属性设置之外的特殊设置和格式化任务。

图形触发器不像在Forms中要由特定进程进行设定排序和命名。可以通过创造过程来创建

China-pub.com 下载

Graphics触发器，并将该触发器与需执行过程的图形对象上的触发事件相联系。例如，当用户单击图8-9所示的散点图上的一个数据值时，希望执行一个动作。

1) 首先，显示如图8-25的Object Properties对话框以选择符号对象。可以使用弹出式菜单，双击对象，或选择主菜单的 Tools |Properties(工具｜属性)菜单项实现。

2) 从Procedure下拉框中选一可用过程，或单击下拉框旁的 New(新建)按钮，在 Program Unit Editor中显示如下代码模板：

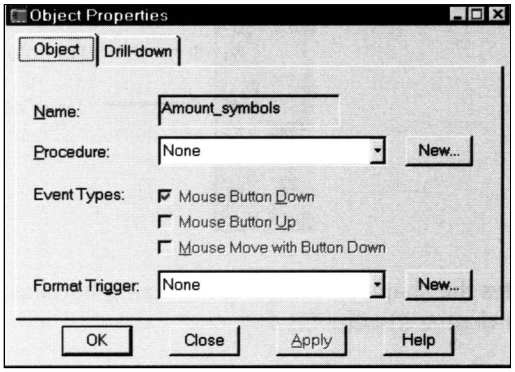


图8-25 Object标签页

```
PROCEDURE OGBUTTONPROC0 (buttonobj IN og_object,
                         hitobj IN og_object,
                         win IN og_window,
                         eventinfo IN og_event) IS
BEGIN

END;
```

在代码模板的BEGIN与END之间可以加入自己的PL/SQL代码，可以按需要改变过程名字。用同样的方法可以建立Format Trigger(格式化触发器)，无论何时，Graphics运行时系统格式化对象时都触发它。

可从不同的属性表中访问不同的触发器事件。既可以使用Tools | Display菜单项打开Display Properties对话框设计Open和Close触发器，也可通过Object Navigator创建Post Execution(过后执行)触发器，该触发器以节点形式出现在查询对象下面。还可用类似的方法设计 Timer(定时器)触发器，首先选择想给要指定的过程的定时器，然后单击它下面的 Timer Procedure(定时器过程)节点。

设计过程系统时，要尽可能考虑通用性，便于日后重用。比如，可创建在多个显示模块中包含的PL/SQL触发器过程库。关于创建在不同显示模块中可重用 PL/SQL的建议查看第10章、第11章和第12章。

在Graphics中编程时，经常需从显示模块中引用对象。第 11章介绍了Oracle Developer提供的一整套扩展功能，利用这些功能可以查找和使用对象标识符或句柄，该章还包括一个用于查找Graphics中各种可用句柄的内部函数表。

注意 当内部子程序调用某一定义时，应提供系统全局变量OG_INCH的相对信息，而不是绝对值。这个变量使用显示器的分辨率，并说明当前显示模式下每英寸布局单位数。因此，可使用下面的PL/SQL代码，设置已存在的矩形aRect的大小(宽2英寸，高4英寸)：

```
aRect.width := 2 * OG_INCH;
aRect.height := 4 * OG_INCH;
```

至此，Oracle Developer三个主要部分的基本内容都介绍完了，本书的下面几章将介绍一些涉及应用程序安全性与重用的高级编程方法。下一部分深入学习 PL/SQL以提高编程能力。

China-pub.com
下载

第9章　禁止非法使用

安全是人类的基本需求。从表面上看，一个数据库应用程序决不会发生使安全受到威胁的事情，但也必须提供一定程度的安全性以防止被破坏。想达到的安全级别正是应用程序采取的安全策略。

9.1　为什么要求安全

安全性的含义是什么？为什么在这上面要花费时间呢？ Oracle或Oracle Developer能自动地保证数据的安全吗？如果系统是通过因特网提供公众使用的，那又如何保护系统以防止非法入侵呢？

下面是由于某些人的故意行为引发的一些事情：

- 不可应用性：系统可能会拒绝合法用户的使用，这可能是因为某些人破坏或使系统过载造成的。如果应用程序是通过因特网使用的，这种可能性更大。
- 不可统计性：由于无法确定数据项的来源，会使系统不能保证应用程序或数据库中数据的准确性或合法性。例如，当一个应用程序通过 Web页获得数据时，世界上的任何人都可以向其输入数据，这就必须有一种方法知道数据的来源。
- 不可控制性：使系统不能控制应用程序的操作，导致不能保证产生的结果的合法性。
- 窃取：有的人会通过伪造或欺骗的方法窃取数据，甚至货币和商品，这也包括窃取智力情报。然而，因特网使所有这一切变得非常容易。

本书并不是一本关于计算机安全性的书，计算机安全是一个具有很多方面复杂问题的广阔领域。在Oracle Developer软件系统中如果没有保护系统来防止被破坏，则可能会发生以上的一些情况。那么，在这方面如何做得更好一些呢？

首先，必须确定来自于故意破坏软件行为的风险。对每一种风险评估失败的可能性及影响，对于那些认为不可接受的风险，采取减少风险的措施，通过这样的分析就可以确定哪些是安全问题，哪些不是。如果认识到很多问题对系统并没有影响，就可以排除掉很多的“安全问题”。决不要陷入这样的陷阱：为了安全，每件事情必须是安全的。

注意　必须确定一些非故意行为破坏系统所带来的一些风险。这种风险的一个例子是：由于查询语句查询非常巨大数量的数据而使系统瘫痪。数据库及应用程序设计应当考虑到这种技术性失败的风险。安全性风险是另外一种猛兽。

其次，应采取的方法是停止那些对软件的未授权使用，并且停止有可能导致非授权使用的授权使用。如果认为必须通过安全措施保证数据的安全性，决不要欺骗自己：使用安全策略可以保证防止某些人使用软件。根据所处理的问题的不同，有可能使得合法的用户也较难访问和使用软件。在采取这样的方法之前，要判断它是否会降低产品的质量。

第三，可能扩展的选择是在取得审查 (auditing)的事实之后提请法庭调查。审查是数据库或应用程序中的一种跟踪事件及活动。建立一种审查跟踪可以帮助及时地调查及处理各种发生的问题，并且可以防止由试探性入侵者所产生的各种问题。

China-pub.com 下载

保证安全性的通用技术是非常广泛的，本章开始所讨论的问题主要着重于 Oracle及Oracle Developer所采用的自主性访问控制和审查。下一节讲述一种更安全的方法，叫做强制访问控制，对于更高安全性要求的应用程序可以使用这种方法。

最后，人们对应用程序的物理访问方法可能决定它真正的安全性。在风险分析及管理计划中必须考虑到物理手段及软件手段。如一个安全性非常敏感的应用程序不能通过公共电话线，当然也不能通过因特网使用。必须限制对运行软件机器的物理访问方法，包括网络连接。

这本书并不要求每个数据库应用程序是绝对安全的，事实上恰恰相反。一个公共的安全模型是限制用户只能访问为完成工作所必须的内容。虽然这样可以产生较好的安全性，但结果是不利于方便使用。如果是为因特网制作程序，这样的方法是不可用的。强加于用户的安全性越少，应用程序将是越易使用的。为满足应用需求牺牲多少易用性及采用多少安全性完全由自己决定。尽可能满足能承受的风险要求，但并不是所有一切。

对安全性访问要求较高的是因特网。如果准备把应用程序发布在因特网上，必须确信应用程序没有违反防火墙的要求，并且不允许黑客 (或任何其他人)进入数据库。 Oracle Developer支持各种防火墙，通过外部防火墙的运行及经过安全的 NET8通道可以访问防火墙另一边的各个服务器。

9.2 自主性访问控制

自主性访问控制(discretionary access control)是控制访问应用程序及数据库的一种能力，它通过给使用应用程序及数据库的主题 (用户及进程)授权的方法实现。这一节概述在 Oracle Developer中可用的各种不同的自主性访问控制方法，给出了标准 SQL安全性控制与Oracle提供的标准之外的安全性控制的区别，并且详细给出了在 Oracle Developer中可用的实现自主性访问控制的不同方法。

注意 下面几节中描述的很多安全性方法是基于服务器的，不是Oracle Developer的内容，然而这些概念很重要，本章在某些细节给出了解释。更详细的内容可以参考关于SQL的书或Oracle文档(参看下面各节中的特殊引用)。

9.2.1 SQL访问控制

ANSI SQL 标准提供了基于Oracle安全系统的最基本的访问控制级别。如果要开发一个使用ODBC的可移植数据库的应用程序，必须使用 ANSI 安全技术加以限制，否则就可以无限制地使用Oracle的各种扩展功能，或任何其他想使用的数据库管理器。

1. 模式及授权

ANSI 安全性方案起始于模式及相对应的授权标识符。在 ANSI SQL中模式(schema)是一个拥有特殊表集、视图集、权限集的对象(视图集及权限集将分别讨论)。任何特殊的表、视图、权限只能属于一个模式。每一个模式有一个名字，称作授权标识符 (authorization identifier)。这个名字也可用于标识一个表，因为不同的模式可以包含具有相同名字的表，也就是说，模式为它所有的对象的名字提供一个范围。在 Oracle中，这些术语等同于用户 (user)及用户名(user name)。Oracle也允许为这些模式提供一个口令(在下面关于Oracle的扩展功能中将会学到更多内容)。

授权标识符是分别访问数据库不同部分的基本方法，应用 SQL的这种特征，根据不同访

China-pub.com
下载

问控制可以把数据库分给各个模式。在模块及模式之间创建关系可以构造复杂的访问控制。例如，在一个中心的模式中创建基本应用程序数据，然后为使用该应用程序的不同人员创建不同的用户，为他们授予合适的访问数据级别，这种方法是十分普遍的。

模式及授权标识符通过把数据库分成各个部分，而不是将数据库看作一个整体的方法，可以实现限制访问数据库中的某一部分对象的安全性目标。

2. 权限

ANSI权限(privilege)用一个特殊的授权标识符来表示对表或视图的操作的分类，共有五种操作(actions)：

- INSERT：允许被授权者在表或可修改的视图中插入新行。
- DELETE：允许被授权者在表或可修改的视图中删除行。
- SELECT：允许被授权者从表或视图中的读数据行。
- UPDATE：带有一个可选择的列表，允许被授权者改变表或修改视图中所指定列的值。
- REFERENCES：带有可选择的列表，允许被授权者在一个完整性约束中参照这些列。

可用GRANT语句定义一个权限，格式如下：

```
GRANT <privileges> ON <table name> TO <grantee> [{, <grantee>}...]
  [WITH GRANT OPTION]
```

ANSI安全系统通过把授权标识符与一个称作模块的抽象对象相联系的方法运行，这个模块包含针对数据库执行 SQL。要执行模块中 SQL语句，必须对授权标识符授予必要的权限。这种方法对数据库中的表及视图提供了一个基本的访问控制级别。

WITH GRANT OPTION子句允许被授权者授权给另一个被授权者，否则只有授权标识符的所有者可以为另一个对象授权。

Oracle及1992 SQL标准增加了一条取消权限的 REVOKE语句，用该语句可选择性地逐级取消被授权者的权限。1992标准还将它扩展加入到模式中的所有对象的权限 (范围、字符集等)，对INSERT权限增加了列表，对各列还可以带 Default说明缺省值。1992标准为新的对象增加了一种新权限USAGE，提供使用范围(domain)、字符集、转换(translation)、校勘(collation)的能力。

使用模式，可以达到如下的安全目标：

- 利用对其他授权标识符的授权，可以限制对特定表、列、视图的访问。
- 利用几种类型的权限，可以限制对某些表及视图的访问种类。

3. 视图

视图是一个看起来像表一样的对象，但事实上它是一个定义于该表的 SELECT语句。视图可以从一个或更多基本表或视图中导出数据，某些视图是不可修改的，如带有连接、分组或表达式的SELECT语句。

为了安全性目的可以使用视图，因为它可以压缩地引用基本表，当使用它时不需要有关基本表的任何权限。也就是说，可以在一个模式中创建基本表，创建该表上的一个视图，并仅把该视图的权限授予其他的授权标识符。

使用视图，可以达到下面的安全目标：

- 通过视图定义中的SELECT列表，可以限制访问一个或几个基本表的某些列。
- 通过视图定义中的WHERE子句，可以限制访问一个或几个基本表的特定行。

China-pub.com 下载

9.2.2 Oracle 访问控制

为了为数据库服务器提供一个完整的自主性访问控制系统， Oracle采取了几种不同方法扩展ANSI SQL标准。

对于Oracle的访问控制及审查 (auditing)功能的详细描述参看 Oracle Server Concepts Manual(Oracle服务器概念指南)。

1. 认证

认证(Authentication)是验证一个用户是否与其声明内容相符的过程，以便保证能够合法使用授予给该用户的权限。正像前节所提到的， Oracle也为授权标识符提供了一套口令系统，在Oracle中称作用户名。 Oracle为授权标识符提供了更进一步的应用，并在 SQL中增加了CREATE USER命令。该命令可以创建一个带有口令的用户 (同时定义了一个不含任何对象的模式)，同时指定缺省及临时表空间、空间限额及配置文件 (下一节讨论表空间及配置文件)。

Oracle提供两套独立的口令机制，第一套假定操作系统提供了一套口令方案，并且对于Oracle会话不使用口令检查。

```
CREATE USER <user name> IDENTIFIED EXTERNALLY
```

第二套方案将口令以加密的形式存在服务器上数据库数据词典中，无论何时启动一个会话都会检查它。使用CREATE USER语句，在任何数据库中可以使用任何一种方案或同时使用两种方案。

```
CREATE USER <user name> IDENTIFIED BY <password>
```

如果用EXTERNALLY，Oracle使用一个标准的前缀，通常是“ OPS$ ”，加在操作系统用户标识符前。因此在使用 CREATE USER命令时，必须在<user name>中加相同的前缀。对于特定系统参看OS_AUTHENT_PREFIX初始化参数。

无论采用哪一种方案，只有经常改变口令才会好用。用户如果不能很经常地改变口令，至少也应每个月改变一次。口令应当是单词或不具有明显意义 (如名字等)的杂乱的符号，并应包括奇异的符号 (如$、#、_等)。人们不应把口令写在黄色的胶纸上并贴在终端上，也不能把口令告诉街上的过路人，或把口令放在电子公告板上。系统验证安全的强壮性取决于使用该系统的用户的可信程度。

当有人启动一个会话时， Oracle 的验证方案完成验证该用户身份的安全目标。

2. 角色

一个Oracle角色(role)是一个权限集合。可以把多个权限授予一个用户，或者把一个权限集合授予一个角色，然后把该角色授予该用户 (或多个用户)。这种方法提供了多个不同用户管理非常复杂的权限组合的能力。

例如， Talbot 雇了一个新的会计师，则仅仅把他或她的用户名字加到 Accounting Role(记帐角色)中即可。如果一个应用程序需要一个新表及在该表上的权限 ,Talbot 仅需改变角色的权限，并且所有具有该角色的用户都会得到正确的权限。

要建立一个角色，必须首先确定安全目标。每一个角色应当是表示某一功能的连贯的权限组。例如，每一个应用程序中都会有一个建立好的唯一角色，它包含运行该应用程序所必须的所有权限。对于一个复杂的应用程序，可能需要划分几个使用该程序的不同用户组，每一个用户组可能会得到不同并且是相互交错的权限集，对每一个用户组可以授予不同的角色。这些应当作为应用程序需求分析的一部分，只有划分好了不同角色，才会比较容易地确定这

China-pub.com
下载

些角色的访问需求。

也可以把角色授予其他角色，这样可以在层次状的角色中，将一个角色与附加的权限或其他角色进行组合。

可以通过在CREATE ROLE语句中带有IDENTIFIED BY分句的方法，对角色实施口令保护。这种方法使得在获得角色的权限之前，用户需要先输入角色的口令激活该角色。这样对角色增加了更进一层的保护，但显然加大了用户注册的难度，只有那些具有重要风险需要验证的事情，才需进行角色口令保护。

Oracle还提供了一个特殊的关键字 PUBLIC，用于表示数据库当前定义的所有用户，该用户组缺省可以访问数据字典的表。通过给 PUBLIC授权，可以把权限授予所有用户，这在有时是非常有用的。还可以把同义词(下一节描述)和数据库链结创建成PUBLIC形的同义词或链接，这意味着所有用户都可以引用它们。

角色应用于安全目地可以更容易地管理复杂的权限集合，可以完成更精细的访问控制，维护这种安全结构只需做很少的工作，这也达到了限制访问的目标，即限制用户只需访问为完成本职工作所必需的数据。

3. 系统及对象权限

Oracle 在服务器安全系统中增加了几种不同种类的权限，新增权限的主要原因是由于Oracle在数据库标准表和视图之外附加了一个模式 (schema)对象扩展集：

- 同义词(Synonym)：任何对象名字的别名。
- 簇(Cluster)：是一种把共享公共信息的多个表组织在一起存储的存储结构，例如与外部关键字完整性约束相关的表。
- 索引(Index)：对表中的数据提供可选择的访问路径的辅助的存储结构。
- 序列(Sequence)：可以用于产生唯一的整数值作为主关键字的对象。
- 过程(Procedure):一个PL/SQL存储过程或函数。
- 触发器(Trigger):与作用在表上的服务器事件相联系的存储 PL/SQL过程，如在UPDATE或INSERT之前或之后的事件。
- 快照(Snapshot):保存从主表(通常是远程数据库上)查询结果的一个表。

系统权限(system privileges)允许管理自己的模式及它的操作，这通过执行特殊对象类型上的特殊操作完成。获得这种权限后，可以使用CREATE、ALTER及DROP命令从模式中增加、改变及删除任何不同类型的对象。在权限上增加关键字 ANY，意味着不仅可以在自己的任何模式，而且可以在不属于自己的模式上实施权限。

除了管理模式对象的权限以外，还有一些特殊的系统权限，这些权限主要是用于数据库管理，但是偶尔也可以把它们授权给应用程序角色，这在表 9-1中解释。

只有当另一个用户给你授予带有ADMIN OPTION的权限或具有GRANT ANY PRIVILEGE权限时，才可以把系统权限授予别的对象。

对象权限(object privilege)是对一个特殊的、现存对象采取某种动作的权限。表 9-2描述了这组权限。

表9-1 Oracle系统权限

权 限	描 述
BECOME USER	允许从任何模式中输入数据

China-pub.com 下载

(续)

权　限	描　述
BACKUP ANY TABLE	允许从任何模式中输出数据
EXECUTE ANY PROCEDURE	允许执行任何函数或过程或引用任何可用的公共程序包
FORCE [ANY] TRANSACTION	对两阶段提交中值得怀疑的事务允许强制执行 COMMIT 或 ROLLBACK
GRANT ANY PRIVILEGE	允许授权给任何系统或对象权限
GRANT ANY ROLE	允许把任何角色授权给任何用户
SELECT/INSERT/UPDATE ANY TABLE	允许查询、插入、修改任何数据库表中的数据
LOCK ANY TABLE	允许对任何数据库加锁
MANAGE TABLESPACE	允许对表空间进行在线及离线备份
READUP	允许用比当前会话高的访问类别查询数据 (看下一节的强制访问控制)
RESTRICTED SESSION	允许在受限服务器访问模式下注册
SELECT ANY SEQUENCE	允许引用顺序值
UNLIMITED TABLESPACE	允许超过给定的空间限制
WRITEDOWN	允许用比当前会话低的访问类别CREATE、ALTER、DROP、INSERT、UPDATE、DELETE对象(参看“强制访问控制”)
WRITEUP	允许用比当前会话高的访问类别CREATE、ALTER、DROP、INSERT、UPDATE、DELETE对象(参看“强制访问控制”)

表9-2　Oracle对象权限

权　限	表	视图	序列	过程、函数、软件包	快照
ALTER	X		X		
DELETE	X	X			
EXECUTE				X	
INDEX	X				
INSERT	X	X			
REFERENCES	X				
SELECT	X	X	X		
UPDATE	X	X			X

自主性访问控制系统假定开始没有权限存在。因此，必须明确地对一个用户或对角色授权，该角色与在解决问题中能够采取行动的某人的用户相对应。

对于给角色及用户赋权限的SQL GRANT和 REVOKE命令语法的详细讨论，参看《Oracle Server SQL language Reference Manual》或有关Oracle SQL的书，如Koch及Loney的Oracle书：《Oracle 8完全参考手册》。

使用Oracle系统及对象权限，可以达到下列安全性目标：

■ 采取给用户或角色授权的方法，可以限制访问各种不同的 Oracle特殊对象。

■ 使用几种类型的权限可以限制某些对象的访问种类。

■ 可以限制对数据库服务器及数据库实际所需的维护能力。

4. 配置文件

Oracle 配置文件(profile)给出了一种控制用户在一次会话中耗费资源的方法。 Oracle建有一个缺省的配置文件，它可以用于所有用户，如果设置初始化参数 RESOURCE_LIMIT为TRUE，就可以定义并且使用附加的配置文件。可以使用 CREATE USER语句为一个新用户指定一个特殊的配置文件，或者用 ALTER USER语句把配置文件指定给一个已存在的用户。

China-pub.com
下载

CREATE PROFILE 语句允许定义一个配置文件，在文件中可以设置限制用户在一次会话中所用到的资源为一个或多个，这些限制在表 9-3 中解释。

表9-3 Oracle配置文件资源设置

资源限制	描 述
SESSIONS_PER_USER	限制用户某一特定的并发会话数。例如，如果允许每个人用相同用户名注册到应用程序，就可以使用这个参数来限制对某一应用程序的访问用户数
CPU_PER_SESSION	限制一次会话可以使用的CPU时间数
CPU_PER_CALL	限制一个特定的分析、运行或取数据的调用可以使用的CPU时间数
CONNECT_TIME	限制一次会话的结束时间
IDLE_TIME	在一定的空闲时间后断开会话(在查询执行期间不计时)
LOGICAL_READS_PER_SESSION	限制会话可以从数据库中读数据块的数量
LOGICAL_READS_PER_CALL	限制在一次分析、运行或取数据调用期间，会话可以从数据库中读的数据块数
COMPOSITE_LIMIT	限制在一个复合单元中一次会话总的耗费
PRIVATE_SGA	限制一个会话可以在SGA中分配为私有空间的总数量

可以使用Oracle的审查功能(在本章的“审查”中讨论)及Oracle的进程管理和跟踪功能来收集不同种类用户的资源信息。然后应当分析与这些资源相关的风险，并且决定采取什么样的策略来限制风险。

对服务器资源的交互使用也会有风险，例如由于资源的耗尽会使应用程序过载或破坏应用程序，用户配置文件可以达到限制这种风险的安全目标。

5. 表空间及空间限额

配置文件允许控制交互作用的资源，但对存储资源又怎样呢？在 CREATE USER语句中给出表空间的使用限额，所以有能力控制允许用户在数据库中使用的存储空间数量。

表空间(CREATE TABLESPACE)是一组操作系统文件及在磁盘上创建表、索引、簇的存储规范。

空间限额是用户可以在一个特定的表空间上创建的空间数量。

使用表空间，可以把数据分布在物理数据库的特定部分。使用不同的 CREATE语句，可以在特定的表空间上创建属于各用户的对象。使用空间限额，可以限制某一用户在一特定的表空间上针对它所创建的对象增加数据的数量。

例如，如下语句在缺省的表空间上创建 Talbot用户并指定了空间限额：

```
CREATE USER Talbot IDENTIFIED BY GeorgeDEFAULT
   TABLESPACE LocalTables
   QUOTA 5M ON LocalTables
   PROFILE CentralAdministration;
```

空间限额达到限制与物理存储资源耗费有关的风险的目的。

9.2.3 客户安全

列的更新控制及菜单访问控制为 Oracle Developer提供了一些其他必需的安全形式。前者允许通过Oracle Developer控制对列的修改，后者使用定制菜单特征允许 Oracle角色激活或关闭菜单项。

注意 这两种特征都依赖于Oracle，如果有数据库移植性的要求，任何一种特征都是相

China-pub.com 下载

关的。

1. 列的更新修改

在第5章的注册过程节详细给出了开始设计一个表单的步骤，有些步骤是设定列安全性的。

块(block)的Update Allowed(允许更新)属性允许控制是否让用户修改块记录，修改一条记录表示要查询该记录，然后改变一个或多个项。如果设置 Update Allowed为False，则用户就只能查询记录但不能改变它们。

项目(item)的Update Allowed属性用来控制是否可以对记录的一个特定列进行修改。如果块属性设置为真，则可以设置一个或多个列属性为 False，以防止用户改变这些列的值。项目的Update If Null(为空时更新)属性可以防止用户对任何值的修改，除非值为 NULL。这意味着用户可以键入值，但是一旦离开该项目之后就不能再改变它的值。例如，在一个工程管理系统中，可以把任务的结束日期设为 Update If Null，当输入一个结束日期以后，表示任务完成，并且系统不允许以后的用户改变该日期。可能有用户不喜欢这种限制的反映，但有时安全性是件不受人感谢的工作。

这种方法达到了这样一个目的，即防止通过表单的特定的块改变表的行或值。可以在设计表单时设置这些属性，也可以在运行时通过调用 PL/SQL触发器改变属性设置。

Oracle Developer通过列安全性(Column Security)方法可以做一些工作。可以设置表单的列安全属性为True，当一个用户注册到该表单时， Oracle Developer在数据字典中查询表单中块的表及列的修改权限，对于没有 UPDATE权限的用户的所有列，它把Update Allowed属性设为False。用这种方法可以利用服务器上设置的权限。

2. 菜单访问控制

如果创建自己的菜单，可以使用 Oracle Developer的菜单定制特征，通过 Oracle安全性角色关掉或删除菜单项。甚至可以使用 Oracle Developer的对话框设置安全性角色，并把角色授权某个特定用户(利用File|Administration|Database Roles菜单项)。不管怎样，在大多数情况下，应使用SQL脚本，或是通过对Oracle数据库的管理接口完成该功能，这样更容易看到发生了什么。在定义角色的同时，应当在数据字典中安装 FRM50_ENABLED_ROLES视图，并且把视图的SELECT权限授予要用安全菜单开发应用程序的所有用户。为了完成这些工作，必须使用Oracle Developer Build和Admin组中的Grant脚本安装Forms所有数据库表。

一旦定义好角色，可以进入 Object Navigator的菜单模块，在菜单属性列表中选择 Module Roles属性，点击More按钮可以打开对话框。现在就可以输入想对该模块执行控制菜单访问的所有角色。在该对话框中，有本章的 9.5.3节“ Talbot应用程序安全性 ”将要创建的角色 (见图9-1)。

注意 这个对话框中包含有一个简单的文本列表，在从数据库的模式中取得角色时它不能给予任何帮助，当设置该列表时，可以先写下所有想输入的角色，以免在输入列表时忘记。

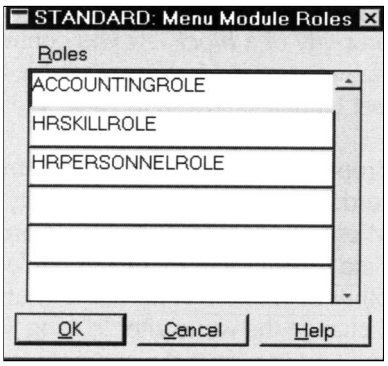


图9-1 Modole Roles对话框

现在，在Object Navigator中找到想进行角色控制的菜

China-pub.com
下载

单项，找到该菜单项属性列表中的Item Roles属性，点击More按钮显示一个对话框，在对话框中可以交互选择激活菜单的角色。当单击鼠标选中一个角色时，它表示被授权该角色的用户可以在菜单条中看到该菜单项是激活的，没有被授权该角色的用户则看到该菜单项是被关掉的。当点击鼠标时按住 SHIFT键，可以连续选择几个项目，当点击鼠标时按住CTRL键可以选择或不选择一个特定的项目，而不管其他各项目的状态(见图9-2)。

图9-2 在对话框中选择激活菜单的角色

Display Without Privilege属性控制Oracle Developer是否可以关掉菜单项或完全删除它，值为 Yes表示关掉它， No表示删除它。现在把菜单属性列表的 Use Security标志设为Yes，选择File|Administration|Compile File重新生成菜单，所有一切就完成了。使用Oracle Developer的这个特点，通过 Oracle 安全特征就可以自动实现安全措施。

注意 如果想开发并测试应用程序而不受菜单的安全性影响，让 Use Security值为No，直到准备发布应用程序为止，到时把它的值改为YES并重新生成菜单模块即可。

9.3 审查

审查(Auditing)是调查一个数据库系统及其用户活动的过程。可以监视数据库的活动，跟踪这些活动直到它们的起源，或收集关于活动的统计信息。也可以记录有关历史信息以确保能够重新构造一个事件链。所记录的信息 (不管信息属于什么类型)叫做审查跟踪(audit trail)。

9.3.1 Oracle中的审查

Oracle提供如下扩展审查功能：

- 语句审查(Statement auditing)：跟踪一个或多个用户某一特殊类型语句的执行。
- 权限审查(Privilege auditing)：跟踪一个或多个用户与某一特定系统权限相关的活动。
- 对象审查(Object auditing)：跟踪数据库所有用户对某一特定的模式中对象所执行的所有语句。

所有这些选择允许把审查跟踪存在表 AUD$中，该表属于SYS用户。可以通过数据字典视图访问这些信息。有关数据字典支持审查的详细信息请看《 Oracle Server Administrator’s Guide》。

审查是一种检查安全策略是否达到预期效果的方法。如果确认某种风险是非常严重的，就可以审查各种可能引起系统失败的活动。如果系统确实失败了，就可以确定引起系统失败的活动序列，或者可以跟踪到引起失败的可能不合法的根源，并且处理这种根源。通过查找审查跟踪也可以隔离各种错误，如果合适，可以审查到各种成功或不成功的语句。当遇到一个问题并且它几乎没有相关信息时怎么办呢？审查可以提供相关信息。

最好懂得审查的影响，可通过研究不同的选项和进行实验，懂得在应用程序中如何应用于安全性问题。

审查可以达到总的安全目标，它可以保证满足其他各种安全目标。

China-pub.com 下载

9.3.2 客户审查跟踪

Oracle可以审查数据库的活动，但它无法扩展到去审查客户应用程序中各种有意义的事件。用户根据自己的安全观点，可能想在应用程序中建立附加的审查。

一个非常广阔的区域就是应用程序审查 (application auditing)。这是把事务历史当作应用程序数据一部分的一种标准办法。如，在一个记帐应用程序中，必须保证能记录所有对数据修改的审查跟踪，在任何一给定时刻，必须能对某一值重新构造出它的正确值，而不管这个值在数据库中已经发生了什么变化。必须能将每一个对象与现实世界的对应物品相联系，如一个货物清单记录必须与货架上的实际货物相对应。所有这些需求在为应用程序设计数据库时必须是明确的。

还可以有一些基于某种特殊安全论点的需求，这叫做论点审查 (issue auditing)。如，可能有一些有关事务的论点对应用程序是非常重要的，想验证一下在所有环境下事务是合法的并且成功，就可以把事务的跟踪存储在分离的审查跟踪中来验证。这些是Oracle提供的大致功能，但它可以应用于应用程序中有意义的事件，而不必用于 Oracle审查已提供支持服务器事件。

一般地，由于审查跟踪的这种直观特性，可能希望把审查及跟踪表封装在独立的数据库模式及过程中，以便客户不必干预审查过程。参看前一节关于模式和授权、认证及过程封装，也可看第13章关于跟踪的进一步讨论，在第 13章可以跟踪SQL执行，可以了解一个应用程序系统在运行时大量的其他信息。

9.4 强制访问控制

强制访问控制(mandatory access control)是一种对于数据库对象实行非常强的访问控制的安全系统，它给每一个对象及会话加一个安全性标号，标号提供了一种分级方案，允许服务器基于分级的级别而不是特定的权限来决定是否授权访问某个对象。

为什么这样的一个方案是必要的呢？在自主性访问控制下，数据仅仅是数据，如果把数据移到另一个地方，数据的权限是由它所在的地方决定的，而不是数据自身。例如，一个用户对某一个表具有SELECT权限，但不带GRANT选项，该用户仍然可以把数据拷贝到另一个表中，这个表是他事先建好的具有相同结构的一个表，并且可以授权其他人访问这个表。有很多类似的情况，限制了自主性访问控制管理安全风险的能力。

强制访问控制通过把安全约束与数据相联系的方法来处理大多数的情况，当拷贝数据时，也同时拷贝了安全约束。可以使用权限完成这些，但用分组方案的安全级别处理起来更容易些。

Trusted Oracle

Oracle提供了强制访问控制方案，使得 Oracle与U.S.B1 TCSEC标准及欧洲ITSEC标准的E3安全级相一致。这是一种可选择产品叫做 Trusted Oracle(委托Oracle)，这种系统只能运行在提供类似特征的操作系统上，这种特征与安全标准一致。例如， Windows和Macintosh不能用做Trusted Oracle的服务器，也不能把应用程序放在WWW(World Wide Web)上，除非浏览器是运行在一种安全的操作系统下。

关于Trusted Oracle中强制访问控制方案的详细信息，请看 Trusted Oracle Server Administrator’s Guide。下面概述一下，它是如何工作的。

China-pub.com
下载

在这种方案中，每个标号(label)有分级部分、类别部分和标记部分。

1. 分级组件

分级(classification)是一个层次级制，它确定被标记信息的敏感性。可以在操作系统级定义分级及顺序，例如：TOP SECRET、SECRET、SENSITIVE、UNCLASSIFIED。

2. 分类组件

分类(category)是对分级的一种划分，它由分级区域组成。即使在分级方案中，也允许限制访问一个特殊的区域，例如，可以让每一个工程具有类别，类别是按工程来划分级的，只有标号中含有工程类别的会话才能看到相关工程的信息。

3. 标记组件

标记(marking)是在一个分级中更详细的一组信息，它在不同环境下与数据相联系。例如，可以把一个特殊的对象标记为 PROPRIETARY，应用程序可以保证打印出所有带有 PROPRIETARY的对象，同时带有一条关于传播的特殊警告。

当注册一个会话时，强制访问控制系统是通过将对象的标号与会话的标号进行比较来工作的，会话的标号是通过操作系统的安全通行证建立的。这种比较是通过支配过程工作的：一个标号的分级如果大于或等于另一个标号的分级，它就可以支配 (dominate)另一个标号，并且它的类别是另一个标号类别的超集。如果一个标号支配另一个标号但不匹配，则它是严格的支配另一个标号。如果是由于类别和分级的区别，使得没有标号能支配其他标号，则标号是不可比较的。安全许可证 (security clearance)是被授权读及写信息的标号范围。只有当标号可以支配对象的标号时，才能读对象，只有当标号匹配了对象的标号时，才能写对象。

与这些约束相关的有几个Oracle系统权限：

- READUP ：允许用严格支配用户标号的标号读数据。
- WRITEUP：允许用严格支配用户标号的标号写数据。
- WRITEDOWN：允许用用户标号严格支配的标号写数据。

关于系统及对象权限的说明及利用系统及对象权限进行自主性访问控制的更多信息，参看9.2节“ 自主性访问控制 ”一节。

总之，如果需要非常的安全，政府或其他顾客一定会要求应用程序与 B1或其他安全标准一致。操作系统及Trusted Oracle的结合，能够利用Oracle Developer去开发这样的程序。注意，不管怎样， Oracle Developer必须运行在可靠的操作系统上以与安全标准相一致。

9.5 保证Talbot系统的安全

作为使用Oracle Developer及Oracle安全性的一个例子，本节从头至尾考虑 Talbot安全策略的开发。这个例子很简单，而一个真正的安全策略将是很复杂的。

9.5.1 Talbot的安全风险

对Talbot计算机操作的风险的分析确认了几种可能的安全风险。

对于Talbot来说，系统入侵及智能窃取的主要风险是不存在的。 Talbot农场的工作站主要分布在农场带锁的办公室里，数据库是物理上分离的，除了农场工作站外，没有人能访问到。通过运行在局域网上的因特网有可能入侵网络，但在非安全区域是可接受的，并且分析表明，即使系统完全失败，影响也是很小的。 Intranet有防火墙保护，它会严格控制通过因特网及调

China-pub.com 下载

制解调器的外部访问。在这样的情况下，风险评估小组决定，每天把备份数据存贮在一个安全位置，并制定一套灾难恢复策略，足以能够处理由于入侵造成系统失败的风险。他们还决定，透露给竞争者信息的影响是很小的，不需要另外的措施保护数据，以防止未授权访问。

因为Talbot使用的是一个不提供安全保护的操作系统，风险评估小组决定主要依靠数据库服务器的安全：即Oracle用户名及口令。

与数据库内容有关的风险包括如下三种可能性：

- 修正或改变Ledger入口有可能导致审查问题，更会导致税务审查及法人审查问题。
- 要求工人保存好自己的地址必须有规章制度。如果一个地址通过广告信息泄露了或缺乏安全措施，则风险主要是规章制度惩罚或对雇员的法律诉讼。
- Talbot维护数据库的经验建议，给每一个人完全访问权可能导致数据库的不准确，进而是不可用的。

风险评估小组考虑了两种管理Ledger修改的可能方法。第一种是双份记录对数据库 Ledger的修改，以维护一个审查跟踪。这种方法的优点是 Ledger总是尽可能正确，而且审查者可以从审查跟踪重新构造修改的历史。第二种可能的方法是，要求改正独立的 Ledger入口，并要求各入口是正确的。这种方法的优点是减少了复杂性，减少了系统开支，并且审查容易 (第二种方法在实际记帐业务中是一种标准方法)。小组决定，不允许修改 Ledger入口，并且要求所有修改必须通过附加的入口。做为一种附加安全性，小组决定把 Ledger表从其他表中分离出来，放在一个带口令保护的模式中。

风险评估小组注意到计帐工作的一部分是在 Person表里维护支出信息，这不会扩展管理地址(后面将做讨论)。

风险评估小组决定把 Ledger表分离放在单独的模式中，以确保它尽可能的不受入侵，只有授权的人事部的人才能访问这些数据。

其他表(Person、Skill、WorkerHasSkill)都放在一个模式中，人事部具有操作访问权，其他人只有查询访问权。

9.5.2 Talbot的Oracle安全性体系结构

图9-3给出了起初的UML类框图，对于不同类别的附加模式信息用容器型的方框表示，每个模式对应一个特定的Oracle用户名和口令。

每个实际的系统用户将会有一个具有不同口令的 Oracle帐号，安全小组建议采用标准的口令维护过程(每个月改变一次口令等)。

与这些要求相对应的角色是直观的：

- Accounting(记帐)：要求在Ledger表中能查询和插入但不能修改或删除，能对 Person表进行查询、插入修改及删除，但不能删除或修改个人的地址入口。
- Address：拥有并管理私人地址表。
- HR Skills：要求能插入、修改及删除工作能力，以及人员和工作能力之间的关系。
- HR Personnel ：要求能对Person表插入、修改及删除信息。

除了这些角色外，还有标准的DBA角色，主要是Talbot用于创建模式、用户、角色、权限等等。

为了创建数据库，Talbot用SQL*Plus以SYSTEM的身份注册到Oracle,并执行如下六个语句：

China-pub.com
下载


```
CREATE USER Human_Resources IDENTIFIED BY FT3R$27SD;
GRANT CONNECT, RESOURCE TO Talbot;
CREATE USER Accounting IDENTIFIED BY TYB##5X$5;
GRANT CONNECT, RESOURCE TO Accounting;
CREATE USER Lodging IDENTIFIED BY M45C$BHI_9Z;
GRANT CONNECT, RESOURCE TO Lodging;
```

然后，Talbot就可以连接到每个帐号，通过运行带有 CREATE SCHEMA语句的脚本为记帐程序创建模式。当创建外键并引用另一个用户创建表时，必须确信在那个用户中已经创建了这些表。因此必须以一个特定的顺序创建所有表，以允许外键定义。另外也可以创建所有表以后，用ALTER TABLE添加约束的语法来增加所有外键。

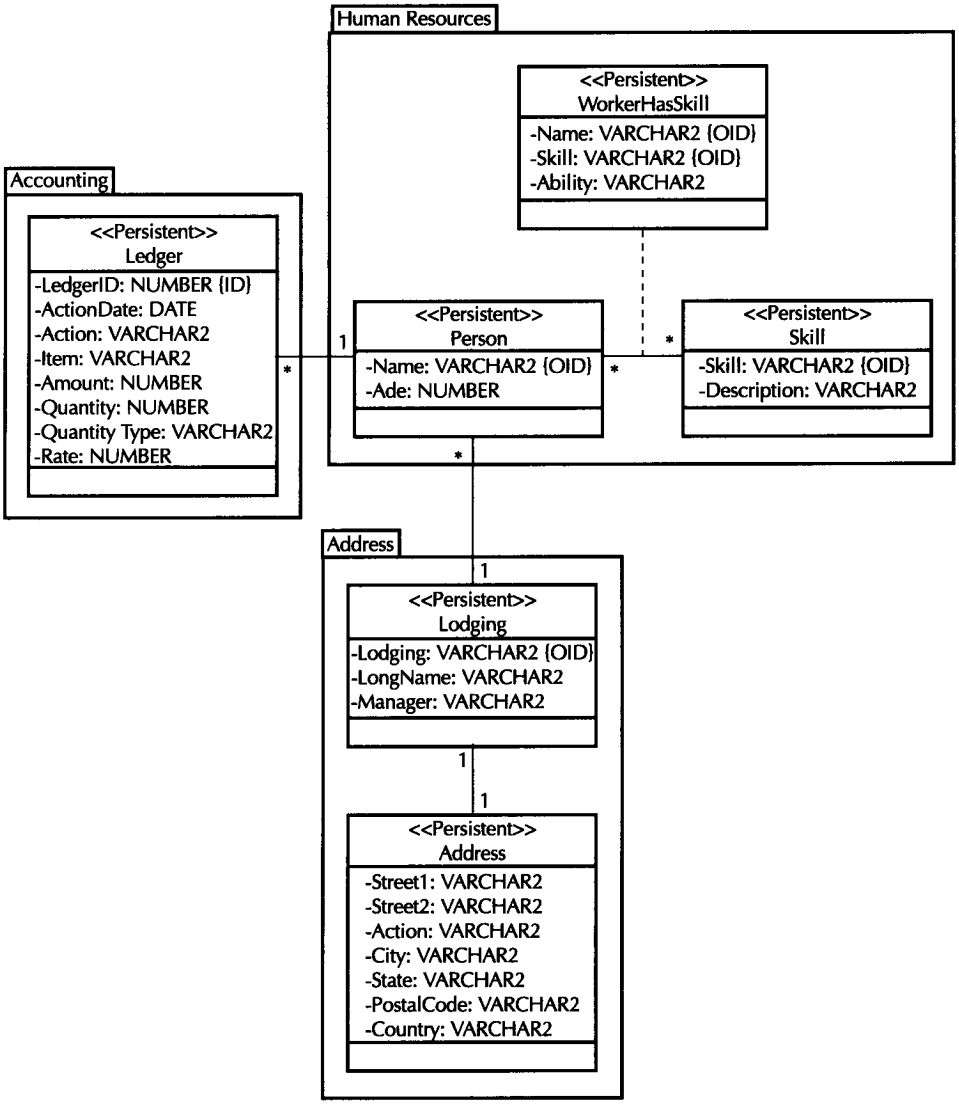


图9-3 Talbot数据库的安全性模式

首先按如下方法创建Lodging模式：

```
CREATE SCHEMA AUTHORIZATION Lodging
```

China-pub.com 下载

```
  CREATE TABLE Lodging (
    Lodging   VARCHAR(15) PRIMARY KEY, /* short name for lodging */
    LongName  VARCHAR(40),             /* complete name */
    Manager   VARCHAR(25),             /* manager's name */
    Address   VARCHAR(30)              /* address of the lodging */);
GRANT REFERENCES ON Lodging to Human_Resources;
```

上面最后一个语句把 REFERENCES权限授予了 Human_Resources用户，允许该用户在Person表中建立参照Lodging表的外键。下面创建Human_Resources模式：

```
CREATE SCHEMA AUTHORIZATION Human_Resources
  CREATE TABLE Skill (
    Skill         VARCHAR(25) PRIMARY KEY, /* name of a capability */
    Description   VARCHAR(80)              /* description of the skill */
  )
  CREATE TABLE Person (
    Name          VARCHAR(25) PRIMARY KEY,       /* worker's name */
    Age           INTEGER,                       /* age in years */
    Lodging       VARCHAR(15) REFERENCES Lodging.Lodging
                       /* reference to short name of lodging */
  )
  CREATE TABLE WorkerHasSkill (
    Name          VARCHAR(25) REFERENCES Person, /* worker's name */
    Skill         VARCHAR(25) REFERENCES Skill,  /* capability name */
    Ability       VARCHAR(15),          /* how skilled is the worker? */
    PRIMARY KEY (Name, Skill)
  );
GRANT REFERENCES ON Person TO Accounting;
```

像前面一样，最后一个语句允许 Accounting用户在Ledger中参照Person表建立外键。还应该注意到，在Person表中的REFERENCES子句，使用了认证标识符 Lodging作为Lodging表名字的前缀。最后，创建 Accounting模式，并一起创建用做Ledger主关键字值的LedgerSequence顺序值：

```
CREATE SEQUENCE LedgerSequence /* sequence numbers for Ledger */;
CREATE SCHEMA AUTHORIZATION Accounting
  CREATE TABLE Ledger (
    LedgerID       INTEGER
                     PRIMARY KEY,/* sequence number, primary key */
    ActionDate     DATE,         /* when */
    Action         VARCHAR(8),   /* bought, sold, paid, received */
    Item           VARCHAR(30),  /* what */
    Quantity       INTEGER,      /* how many */
    QuantityType   VARCHAR(10),  /* quantity: lbs, bushels, etc. */
    Rate           NUMERIC(9,2), /* how much per quantity type  */
    Amount         NUMERIC(9,2), /* rate * quantity */
    Person         VARCHAR(25)
                     REFERENCES Human_Resources.Person /* who */
  );
```

当所有用户及表创建以后，现在就可以设置角色的权限了。 SYSTEM用户是DBA，已经具有所有的权限。连接到SYSTEM并创建各角色。

```
CREATE ROLE AccountingRole;
CREATE ROLE  HRSkillRole;
CREATE ROLE HRPersonnelRole;
```

现在连接到Accounting用户，把Ledger表授权予Accounting角色：

```
GRANT SELECT, INSERT ON Ledger TO AccountingRole;
```

连接到Human_Resources用户，对Person、Skill及WorkerHasSkill表授权限。

China-pub.com
下载


```
GRANT SELECT, INSERT, UPDATE, DELETE ON Person to AccountingRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON Skill to HRSkillRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON WorkerHasSkill to HRSkillRole;
GRANT SELECT, INSERT, UPDATE, DELETE ON Person to HRPersonnelRole;
```

注意 自主性访问控制不能用权限来强行限制Accounting角色修改有地址的用户，而必须用数据库触发器来实行这种约束。也可以用创建一个独立的视图，用WHERE子句去除掉那些数据行的方法来实现这些。然而这种方法不允许应用程序看到那些名字，这与要求查询到所有名字的其他需求相违备。也可以用这个视图进行修改而用基本表进行查询，但这要告诉Oracle Developer数据块如何工作，是非常难的。可能需要两个独立的块，一个用于查询目的，另一个用于查询/修改目的。

创建了所有的角色并授予了所需要的权限以后，现在就可以创建单个用户并授权他们角色。例如为了建立记帐用户和全体人员的用户，用 SQL*PLUS连接SYSTEM，执行下列语句：

```
CREATE USER User1 IDENTIFIED BY JU8##NMY78$;
GRANT CONNECT TO User1;
GRANT AccountingRole TO User1;
GRANT SELECT ON FRM50_ENABLED_ROLES TO User1;
CREATE USER User2 IDENTIFIED BY H#17_TH$5;
GRANT CONNECT TO User2;
GRANT HRPersonnelRole TO User2;
GRANT SELECT ON FRM50_ENABLED_ROLES TO User2;
```

提示 也可以用Oracle企业管理员程序包中的Security Manager，使用一个友好的用户界面创建用户及授权。

注意 给用户授予FRM50_ENABLED_ROLES的SELECT权限是必要的，以便能用Oracle Developer的菜单安全性特征。没有此授权，会看到FRM-10256这样的Forms错误信息：“ 用户没有被授权运行Form Builder菜单 ”。这里假定已经从Administration程序组安装了表单的数据库表，这些表也是必需的。

9.5.3 Talbot应用程序安全性

Ledger表单必须满足要求，不能修改或删除 Ledger表中数据行。 Oracle数据库通过AccountingRole安全角色执行这种限制。如果修改Ledger表中的值或删除一行，当向数据库保存所做的修改时会出现错误，因为SQL语句的执行失败了。

尽管这样解决了问题，但这并不怎么好。接口友好的原则是，当出现错误时要立即给出反馈。而按照数据库的安全性，只有在存盘时而不是在实际改变值或删除记录时给出反馈。为了在程序接口中实行立即约束，需要在应用程序中放一些额外的代码。

为了实现这种方法，所需要做的就是在 Ledger块中关掉Update Allowed和Delete Allowed属性。这将停止用户对从数据库查询到的值的修改或删除，并且会立即给出一个标准的 Forms错误信息反馈。

注意 可以把模块中Column Security标志设为True，但在这个例子中，用块的Update Allowed标志似乎更直接。产生错误提示的一种选择就是关掉删除记录的菜单及键盘操作方法(即Record | Remove菜单项及标准的 [Delete Record] 按键)。这样从根本上不允许执行，将产生一个较好的接口，但是涉及到对工作(创建定制的菜单等)的影响似乎要高

China-pub.com 下载

于所带来的好处。如果已经有自己的菜单及这样菜单项的按键定义，应当修改它们。

在开发其他应用程序时，在基于 Oracle角色的不同应用程序中， Talbot可能会用菜单安全系统配置各菜单。如，在Ledger表单中可能会通过AccountingRole安全角色允许数据操作菜单项(如创建记录或删除记录)，通过其他角色关闭菜单项。也可能允许 AccountingRole角色的用户查看Ledger总结报告，而其他用户不允许。关于设置菜单安全性的例子及细节参看 9.2.3节中“菜单访问控制”。

总之， Oracle Developer协同Oracle 或其他DBMS一起工作，会提供强有力的安全工具。应用这些工具要花一些心思并仔细准备，但完成后可以安然入睡，因为知道应用程序已经达到了所能做到的安全级别要求。

China-pub.com
下载

第10章　高级重用设计

这是第二部分的最后一章，处理与高级应用程序设计有关的几个问题。当掌握了触发器及进程的使用，懂得了表单、报表、图形的基本操作以后，就要准备优化结构。 Oracle Developer提供了几种工具，有助于使应用程序更容易适应环境的变化及多个不同用户的要求。

10.1　再循环：每个人都应做到

可重用性(reusability)是指在不同背景中使用对象的能力。通过减少为每一个所遇到的新需求必须编写代码的数量，直接转换成了增加的生产能力。在新的应用程序中，可以重新使用的代码越多，则为编写该程序所花费的时间、心血及经费就越少。代码重用使得应用程序更加一致，更加容易使用及维护。

如果设计的对象要用于最佳的应用程序，则要求开发组使用这种特征，使得对象在更大程度上可以重用。这其中一部分是销售的工作：对象必须有文档及“公共关系材料”，才能让开发组知道该对象能为他们做什么。其他的工作是使得对象尽可能地使用方便，并给出在特定系统下工作的约束条件，确定建立对象和测试对象的费用。下面各节提供一些工具，用来增进全部Oracle Developer代码的柔韧性。

提示　重使用其他工作组的代码也会有好处，例如离架式的Java Bean和ActiveX组件，你可以购买而不用建立它们。第15章详细讨论关于Oracle Developer提供的把离架式组件综合进应用程序的技术。对于公司内部开发的代码建立一种重用的习惯，这与做出建立还是购买的决定同样重要。

Oracle Developer最大的好处之一就是它的基于对象的特点。每个应用程序是由一个模式构成的，并且Oracle Developer使得重用它们非常容易，因此它鼓励你建立兼容的对象。在任何软件系统中，都可以找到使对象不兼容的方法，并且有些对象是本质上不兼容的，不兼容的原因超出了控制范围。 Oracle Developer提供了容易重用代码的工具，如果遵守下一节的建议，就能提高在不同程序中重用对象的能力。

下一节概述介绍了主要用于重用的 Oracle Developer的元素，也可以查看第12章，在不同模块中将PL/SQL函数和过程打包以便重用的方法。

10.1.1　模块

Oracle Developer主要围绕几个对象构造它的操作，这些对象已在练习中看到了， Oracle Developer的模块有：

- 表单(Form)：含有数据块、记录和数据项的模块，用于在表单中表示一个交互数据库应用程序的数据。
- 菜单(Menu)：这个模块包含表示一系列的命令动作接口的菜单和菜单项。
- 报表(Report)：包含数据模型和报表布局的模块，报表可以从数据库中的数据产生。

China-pub.com 下载

- 显示(Display)：这个模块包含一个或多个图表或其他的图形，一个或多个查询，用于表示数据库中数据的一个图形显示。
- 外部查询(External query)：用Report Builder建立并可以在其他模块中引用的查询。
- 程序库(Library)：用Form、Report或Graphics Builder建立的、可以在其他模块中引用的PL/SQL代码库。

在设计模块时，应当考虑设计成可重用的应用程序组件，而不要仅仅作为独立的应用程序。如果以这种方式设计组件，就可以在其他的应用程序中重用它们，从而在开发生命周期中提高可重用性。

模块可重用性的关键是尽可能降低模块与其环境的耦合，耦合 (coupling)是一个模块使用模块外的数据及操作的程度。这里给出降低模块耦合的一些指南。

如果可能的话，不要使用模块外的变量，如全局变量或其他模块的变量。使用这样的变量越多，在不同背景中能重用该模块的可能性越小。很难预料其他模块是如何使用共享变量的，当与其他模块交叉使用时有可能会产生预想不到的奇怪事情。使用全局变量的一个好方法是创建一个输出变量对象的程序包。在引用附加在模块上的程序包时， Oracle Developer会在该会话期间把程序包装进内存，这意味着可以在模块的任何地方使用这些变量，但在模块外面却不能访问它。这些变量也可以做为一种使用全局变量参数的方法，并且在某些条件下访问的速度会比较快。第13章有关于使用程序包的详细内容。

尽可能把与应用程序有关或与平台有关的组件分离放在过程或程序库里，以此来降低模块与特定应用程序或平台相关的耦合。例如，在特定应用程序中仅需在屏幕上放一确定的文本，可以把它做成从进程触发器调用的过程。如果需要特定应用程序的帮助，就将它放在PL/SQL库中，并为不同的应用程序设定开关。

避免将样本文本(难以编码进画布、报表、显示图表中的文本)与特定应用程序的对象捆帮在一起，在样本文本中减少使用题目，尤其是帮助文本。使用通用的术语，尽可能使用数据库表或列的名字，而不要使用与应用程序版本有关的名字。当它确实不能工作时，应该总是能够修改它。对于某些特定应用程序需要的部分，可以做工作对它进行参数化。

注意 显示在表单中的大多数文本用于组成各个域的提示，Oracle Developer Forms提供了包含这样文本的提示属性。而且，Oracle Translation Builder提供了管理国际化问题的工具。除此之外的事情就是用其他的文本告诉用户干什么，或划分各个项目的组，应当减少使用这样的文本。

如果使用很多PL/SQL过程和函数，应该用较高的集成度把这些过程和函数集成到程序包或库中。用分层和子系统的方法设计模块，可以减少相互之间的连接。由于不合适的结构，使各层之间的连接增多，使得降低模块的耦合以利于在其他地方重用的可能性变小。对于开发者和设计者来说，在各层内所隐藏的模块功能越多，越容易理解模块的功能。但最高层将需了解隐藏在各个模块之间的复杂的相互作用。

10.1.2 模板和组件

如果由于模块针对应用程序太特殊而不能重用，那至少可以把不同的元素抽象成模板而从中取得一些用途。模板 (template)是一个对象，它用基本选项或内容来格式化成一个模块。如果有标准或开发指南，在建模块时把它们建成可以重用的模板。例如，可以创建表单、报

China-pub.com
下载

表或显示模块，让它们具有代表所需标准和指南的所有标准设置。然后，开发小组可以拷贝这个模块作为一个模板，再开始开发他们自己的表单。对于报表和图表，可以开发特殊的模板对象，用结构化的方法表示缺省信息。这一节将详细讨论这方面及其他更多的内容。

模块和模板是Oracle Developer的基本构造块，可重用的模块越多，并且可用的标准模板越多，开发过程将会越高效。

1. 表单模板

与Report和Graphic组件不一样，Oracle Developer Forms并未提供明显的模板(参看下一节“报表模板”和“图形模板”)。Form Builder的欢迎窗口允许你基于另一个表单建立一个新表单。单击Welcome对话框上的“Build a New Form Based on a Template”(基于模板建立一个新表单)单选按钮后会显示出“Open Form”对话框,提示选择一个FMB文件作为模板，Oracle Developer会把所选择的表单拷贝为Form Builder中的新表单。

注意 除了启动Form Builder以外，没有其他方法可以到达这个对话框中。这是唯一可用的、有限的Forms模板特征。也可以使用通常的Open工具打开模板表单，然后使用菜单项中的File|Save As把它保存成另一个名字。

应当建立一个标准表单，它带有想要标准化的对象。然后，可以更高效地用 Data Block Wizard和Layout Wizard建立数据块，并填进画布中，表 10-1给出了这些标准对象可能的使用方法。

表10-1 使用Forms Template对象

对 象	建议使用的方法
Form(表单)	Property Palette中的标准表单设置，如Coordinate System/信息、菜单模块及选项设置
Form trigger(表单触发器)	用When-New-Form-Instance设置标准窗口大小及其他初始化PL/SQL
Alerts(警报)	标准错误和警告提示
Attached libraries(附属库)	多个表单共享的标准程序库
Canvases(画布)	合适的标尺及布局设置的标准画布
Parameters(参数)	每个表单使用的标准参数
Pop-up menus(弹出菜单)	所有应用程序共享的标准式菜单
Program units(程序单元)	特定于表单的标准程序单元，例如从When-New-Form-Instance触发器中调用的初始化过程，或发送给客户信息来支持有关表单问题的标准过程
Reports(报表)	所有表单共享的标准报表
Windows(窗口)	标准窗口和对话框

2. 报表模板

在一个报表中三次嵌进单位标志和名字，你可能会开始怀疑有没有更好的方法。有的，这就是报表模板。第4章在Talbot .TDF标准报表模板中演示了它的用处，这一节要讲述这样的模板是怎么建立的。

一个报表模板是一个特殊的文件，它包含提示文本和图形，包含使用字符、数值和日期不同数据类型的样本域。在 Report Wizard模板屏幕中让你指定重使用哪个模板。用模板提示文本和域创建的新报表，构成了初始的原型。然后就可以在 Live Previewer中处理该原型，以完全满足报表的要求。使用模板，使得报表中不需变化的部分标准化了，节约了重新做所有标准元素的时间。这意味着编制各种格式报表的效率实际提高了。

China-pub.com 下载

Oracle Developer装载了很多标准模板，很遗憾，这不是一些“通用”的模板，而更像一些示例文件，因为在它们中嵌入了很多公司的演示图像和文本。这些标准模板会自动出现在Report Wizard的列表中，可以把这些作为模板的基础，但在使用它们之前需要做一些修改。

1) 要打开一个标准模板，用File| Open显示Open对话框(见图10-1)。

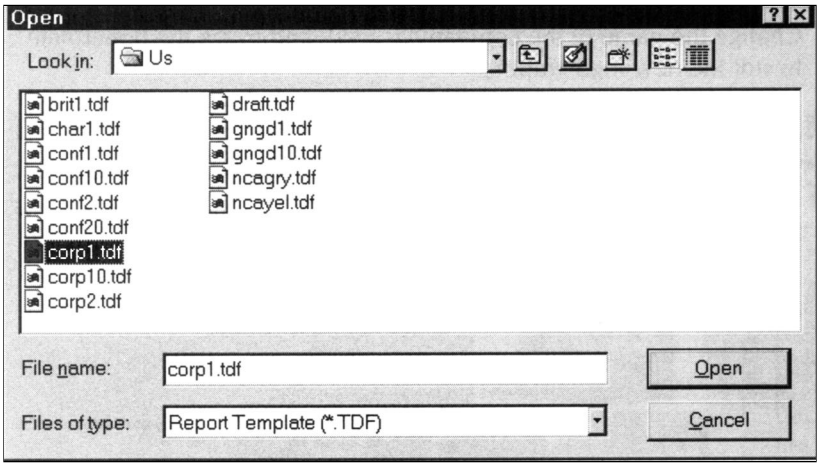


图10-1 Open对话框

2) 用下拉列表把Files of Type(文件类型)域设成Report Templates(报表模板)(*.TDF)类型。

3) 导航到模板目录(通常是在Oracle_Home下的一个Admin文件夹和它的子文件夹)，选择想打开的文件(如CORP1文件)。

模板文件在Template Editor中打开，模板不能使用Live Previewer，因为模板中看不到数据，仅仅看到数据类型(见图10-2)。

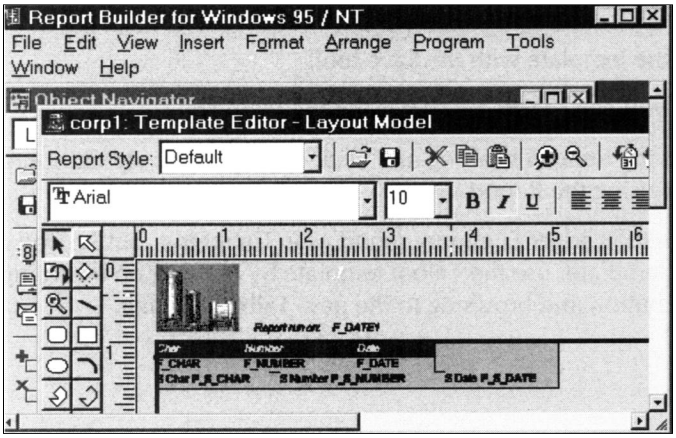


图10-2 Template Editor

如果查看一下Object Navigator，就会看到在模板标题下有了一个新的模块CORP1。CORP1模板是一个有点奇特的布局，有一个位图标志、一个公司名字及一些预先的格式化和布局设置。要使用它，需要替换图像和公司名字。

4) 如果想用一个不同的名字保存这一新的模板，而不愿替换标准模板，可以用 File |Save As完成。

China-pub.com
下载

5) 删除已存在的图像并用自己的标志替换它，修改提示文本，在这个例子里是替换Talbot’s Farm的标志。

6) 改变公司的名字，移动周围的提示文本以适合新的标志图像 (见图10-3)。

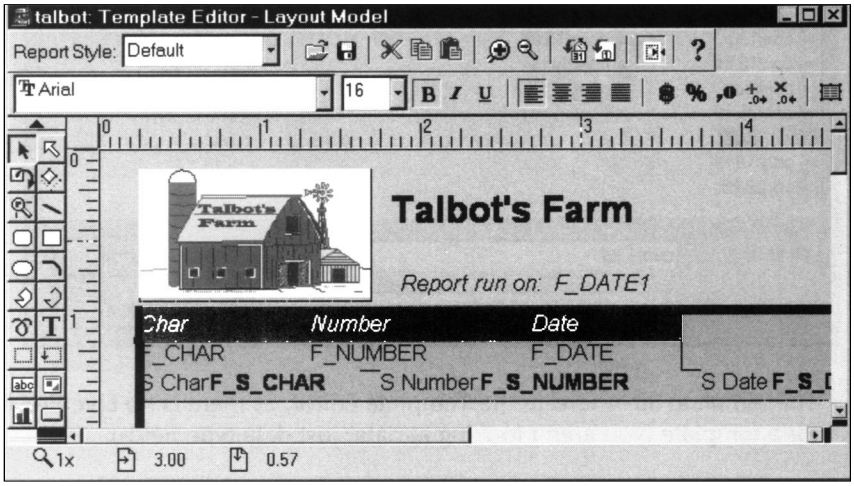


图10-3 改变公司名字、移动提示文本后的Template Editor

7) 用Save工具保存模板。然后可以用模板建立报表。

8) 选择Reports标题并单击Create工具，用Report Wizard创建一个新的报表。

9) 创建第4章中的Ledger Summary报表，但不用Draft模板，而使用Talbot模板，这可通过单击Template文件的单选按钮并浏览到新的Talbot .TDF文件。

注意 Report Wizard中的预定义模板是建立在系统内的，无法往里增加或改变所选文件的名字。另外，也不能改变演示报表的Wizard细小的图形，因为这些是基于模板目录下的一些特殊位图。最好完全不管这些预定义的文件，使用Template文件单选按钮去选择一个特定的文件，而不是修改这些标准模板。

Template Editor可以像报表的Layout Editor那样查看模板的主体、边界、标题和尾部。当查看模板的主体时则不像 Layout Editor，不能画域、添加图形或做任何事情，只能显示缺省部分框架的Property Palette。如果单击Margin、Header或Trailer工具，可以看到各种画图和格式化工具出现，表明可以按照所希望的那样画样本文本或增加域。模板允许定制边界、页头和页尾，但不允许定制布局主体的样本文本或域。这就是从模板创建一个报表时， Report Wizard所做的工作。

在Object Navigator中查看模板的结构时，如图 10-4所示，会看到在模板中有五个主要部分：

- Data Model(数据模型)：报表的系统和用户参数
- Layout Model(布局模型)：报表的主体和边界
- Report Trigger(报表触发器)： Before Report触发器、 After Report触发器、 Between Pages触发器、 Before Parameter Form触发器和After Parameter Form 触发器。
- Program Unit(程序单元)：报表的PL/SQL程序单元
- Attached Library(附属程序库)：在报表、程序单元中想使用的 PL/SQL程序库。

China-pub.com 下载

利用Data Model部分，可以设置各种系统参数的缺省值，如DESFORMAT或ORIENTATION。例如，如果想缺省设置所有报表都发送到打印机而不是 Previewer，就可以把DESFORMAT设成Printer而不是Previewer，也可以增加标准用户参数。

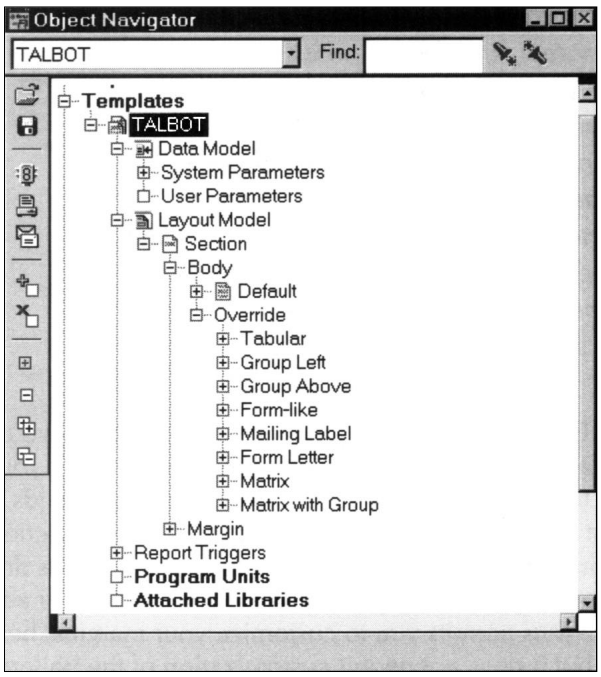


图10-4 Object Navigator中报表模板的层次结构

应该用Template Editor的可视化编辑器和画图工具来修改 Margin对象。其Body部分包含一个可部署的对象阵列，可用于定制不同种类的报表。图 10-5给出了在Body部分中对象的层次。第一个对象是 Default section，用于设计报表主体部分的基本设置，它由 Frames(框架)、Field Labels | Headings(域标注|标题)、Fields(域)、Summary Labels(汇总标签)和Summaries组成。使用这些设置，可以控制报表每个方面的实际缺省外观。

第二个对象是Override，它包含一组覆盖Default区各种设置的对象。这些对象是针对某些特定报表类型的。每一个对象对应着一种报表类型，如 Tabular报表、Mailing Label报表或者Matrix报表。每一个对象都依次包含着报表中不同区域的对象，每一个最终都会包含有类似在Default区中所列的五个对象。对于每种报表中每个特定的区域，都可以使用这些对象覆盖掉在Default区中的缺省设置。

注意 在模板中无法为一个或多个类型的域说明一个缺省的格式掩码，所能设定的只是它们的合法性。

最后，可以把程序库和程序单元附加在模板上，以便每个报表都可以访问它们。这样则允许建立标准的子程序和程序包，任意放在程序库里或模块中，并且在所建立的任何报表中可自动地使用它们，而不用附加或引入它们。

如果花费一些时间为标准报表环境开发些报表模板，就会发现随着时间的逝去，生产力引人注目地提高了。基于布局，不用做过多的格式变化就能快速地建立报表。

China-pub.com
下载

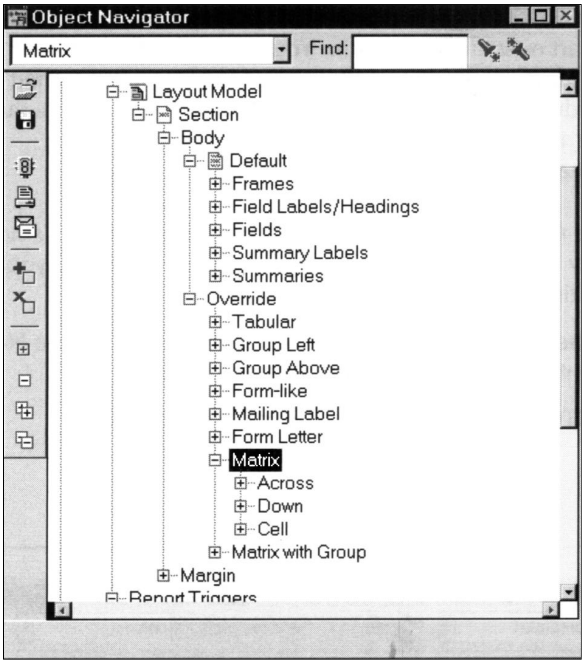


图10-5 Object Navigator中Body对象的层次结构

3. 图形模板

如果开始广泛地使用 Graphics Builder，就会发现一遍又一遍地制做相同种类的图表，也可能会发现所建立的很多图表看起来是一样的，例如下钻图表 (drill-down chart)或多重图表(multiple-chart)。代替个别定制每个图表，可以建立一个定制的图表模板，用来构造所有有类似特征的图表。

建立图表模板与建立图表一样容易。

1) 启动Graphics Builder，打开一个显示窗口，并找到Object Navigator中的模板目录(它正好位于显示的Layout目录下面)。

2) 选择目录并单击 Create工具，或使用 Navigator|Create菜单项创建一个新的模板(template0)。

图10-6显示Chart Templete Editor的内容。

Editor中显示的是框架编辑器及它的框架模板 (frame template)，模板中包含标题、坐标轴标记及刻度位置，可用于设置列式图表。使用 Type菜单，可以从多于 50个的缺省模板中选择任一个(模板有折线图、列式图、条形图、饼图、 Gantt图、Scatter图或表)。

在Editor中可以移动或改变任何框架元素。例如，可选择所有元素，并且把字体改成 Arial 10 Bold而不用系统字体。双击某个元素，可以设置不同的参数使图表框架变成想要的任何形式。

在每个图表模板中只能有一个框架模板，但可以有多个域模板。一个域模板 (field template)是图表中一部分数据显示特征的规范，可以把每个不同的域模板与查询中的不同列相联系，用特定的样式显示每列中的数据。允许设置一组标准线条和填充颜色、字体或其他数据元素的显示特征。

如图10-7所示是一个列式图的缺省域模板。

China-pub.com 下载

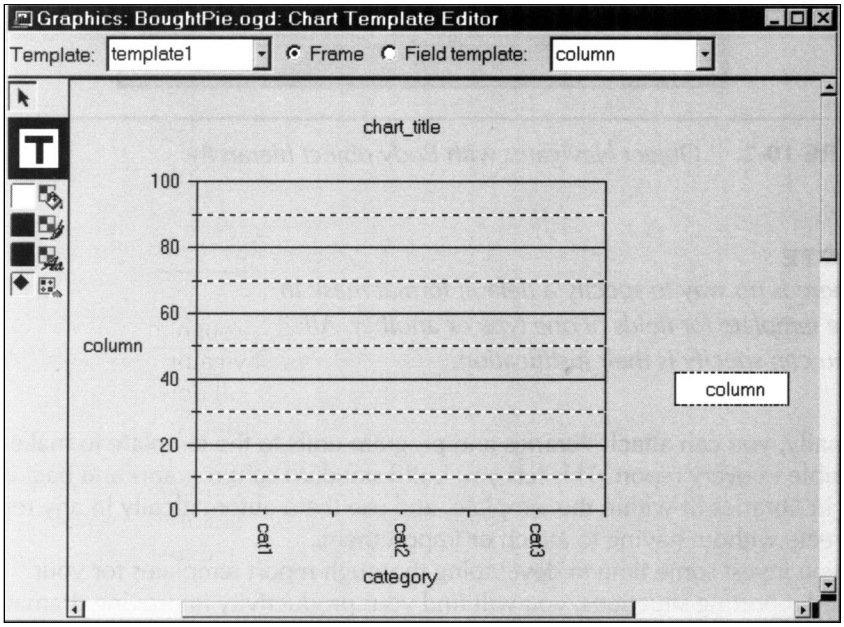


图10-6 带框架模板的Chart Templete Editor

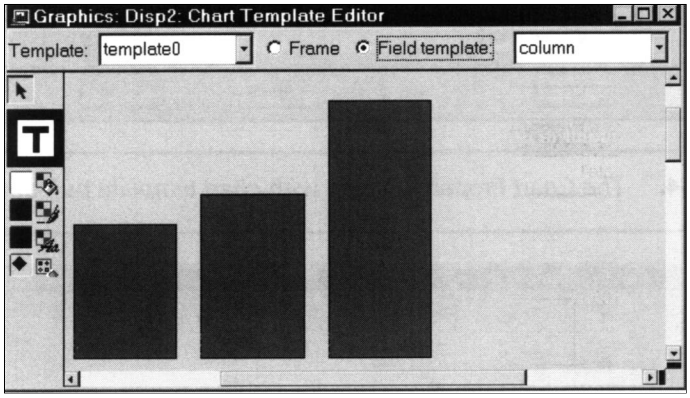


图10-7 一个列式图的缺省域模板

可以通过Chart Property Palette将一个特定的图表模板指定给图表。双击 Object Navigator上的图表对象，在Chart标签页(图10-8)底部的Template域可下拉选择可用的图表模板列表，当单击OK后会根据模板重新构造图表。

可以通过Chart Property Palette的Values标签(图10-9)，将一个特殊的域模板指定给某一列。选择要想格式化的列，然后从可用的域模板列表中选择一个并赋给想设定的列。

如果熟悉Oracle Developer的这种模板能力，就会发现根据图形标准开发大量的图形是非常容易的。开发模板的目的就是为了减少制作一个特定图表所需的常规劳动的数量，制做时仅完成需要变化的部分(如Talbot饼图中的Other饼片)。像字体和颜色之类的应该是标准的，不需要一个图表一个图表地去确定。

要最大限度地使用模板，应当把它们的输出放在独立的模板文件中。

1) 在Object Navigator中选择想输出的模板。

China-pub.com
下载

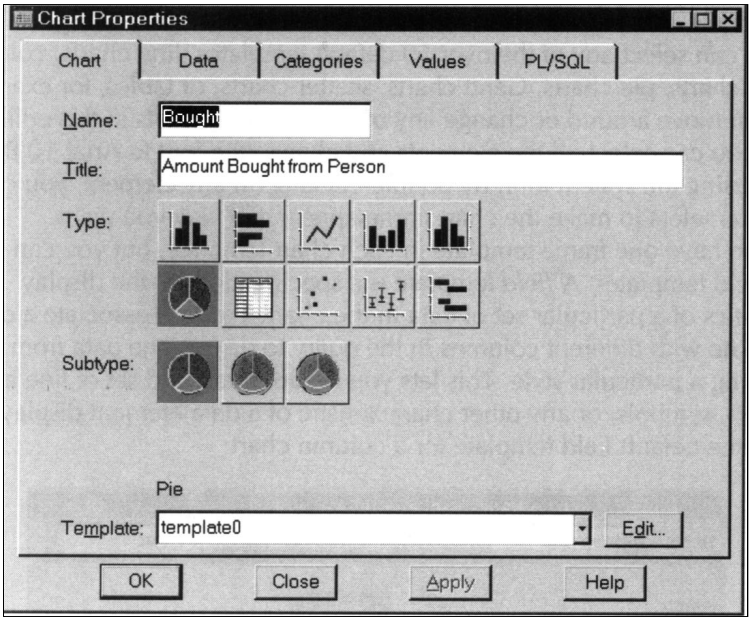


图10-8 Chart Property Palette中的图表模板下拉列表

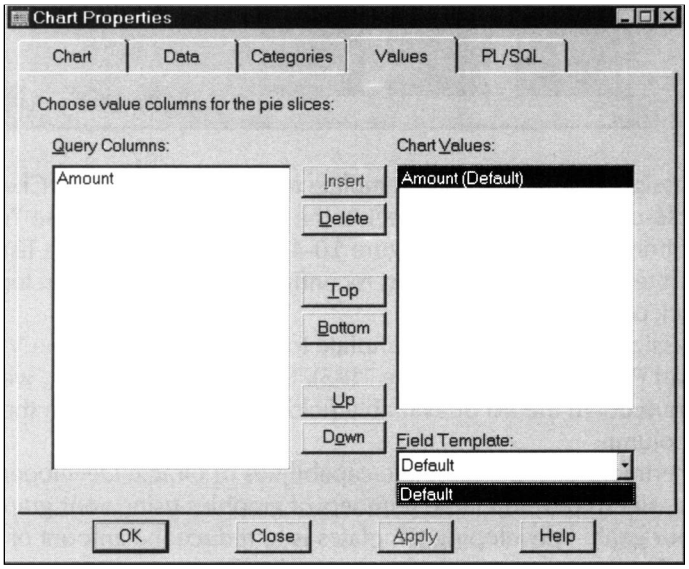


图10-9 Chart Property Palette中的域模板下拉列表

2) 选择File |Export |Template菜单项，会显示Export对话框(见图10-10)。

3) 输入文件名(如“ PIE.CHT ”)。

4) 点击OK，将模板保存在某个中央目录中。

5) 当想在另一个显示窗口中使用模板时，将模板引入显示窗口，选择显示窗口下面的Templates节点，或者显示Chart Property Palette并选择File |Import |Template，显示Import对话框，这与Export对话框非常相似。

6) 找到文件并单击OK，引入模板。

China-pub.com 下载

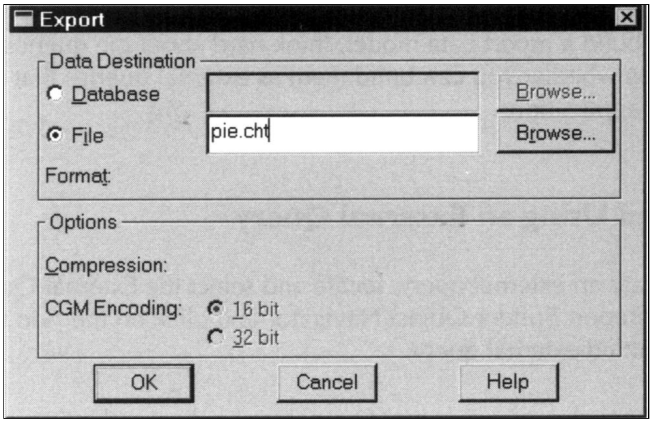


图10-10 Export对话框

10.1.3 外部查询和Query Builder

外部查询(external query)是一个包含SQL SELECT语句的文本文件。可以在任何报表的数据模型中引用该外部查询。把查询单独分成一个模块，使得不同的报表可重用该查询。

注意 Oracle Developer文档一直把查询看做是ANSI标准SELECT，但Report不强调这种对SQL的限制。Oracle 中的注释及发表的例子中的其他元素的出现，表明查询语句可以是目标数据库能够理解和处理的任何SQL查询。

当建立报表数据模型时，应好好考虑一下要建立的查询，看是否把它建成可在其他应用程序有用的外部查询。

1. 创建并使用外部查询

1) 要创建一个外部查询，在Report Builder Object Navigator中确定并选择External Query标题，单击Add工具创建一个Untitled外部查询。

2) 双击External Query，显示刚创建的查询的External Query Definition Editor，见图10-11。

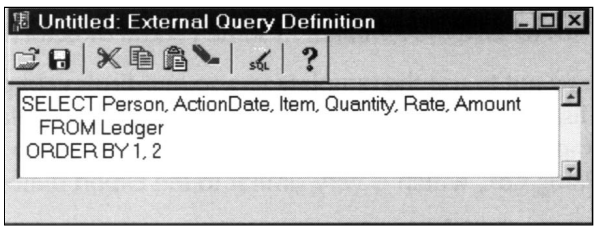


图10-11 External Query Definition Editor

3) 输入查询语句后，关闭Editor，并在Object Navigator中选择UNTITLED对象。

4) 点击Save工具，作为文件保存，并给出文件扩展名 .SQL，如“ LEDGER.SQL ”。

5) 使用Property Palette增加查询，找到Query图标并双击它。

6)点击External Query Source File属性的More按钮，或双击该属性，会显示一个标准的Open对话框，通过它可以选择Ledger.SQL，然后就可以看到带有语句的SQL Query Statement对话框(见图10-12)。

China-pub.com
下载

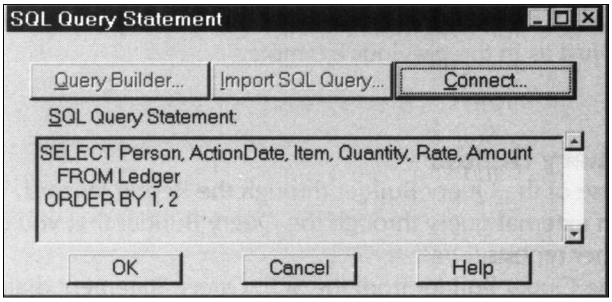


图10-12 SQL Query Statement对话框

注意 另一种方法是在SQL Query Statement属性中点击More按钮,直接打开对话框并点击Import SQL Query按钮打开Open对话框。

7) 要使用Report Wizard，用Tools |Report Wizard菜单项打开Wizard,然后选择Data标签页面。

8) 像前面一样用Import SQL Query按钮打开Ledger .SQL文件。

9) 点击Finish。

注意 使用查询文件名，取决于在注册变量REPORTS60_PATH设定的存贮文件的目录。这样就可以把文件移到不同的文件系统和不同的平台，而不用重新设置Query Property Palette入口。一旦使查询引用一个外部查询，即使它以正常方式显示的，也不能在Property Palette中编辑查询了。要编辑它，必须选择外部查询对象并通过双击它的图标。当点击OK释放Property Palette时，Oracle Developer扫描查询语句并报告所有问题。

定义好了外部查询，就可以在任何报表中引用它。例如，要想创建基于 Ledger查询的一个新报表，但新报表具有不同的布局、不同的求和和分组以及其他等等，就可以建立新的数据模型，并和前面的例子一样在Query Property Palette中引用外部查询。

2. 使用Query Builder

可以通过Report Wizard来利用Query Builder。这个界面允许用Query Builder建立外部查询，并保存它以便让其他报表重使用。

可以从SQL Query Statement对话框进入Query Builder，或者从Report Wizard Data页面进入，就像外部查询一样。不用单击 Import SQL Query，而是单击Query Builder按钮，Query Builder允许以绘画的方式建立SQL查询。图10-13显示了Query Builder中的Ledger查询，当单击OK时，Query Builder创建SQL SELECT语句,并把它放进SQL窗口中。

注意 就我个人而言，我发现通过写SQL来开发查询要比用Query Builder或其他图形工具容易得多，尤其是复杂的查询，用工具必须要进行拖拉、放置和描绘。不熟悉 SQL或者对数据库的模式不是了如指掌的开发者，可能发现用 Query Builder比编写实际的SQL容易。然而，如果要开发接近中等规模的客户 /服务器或因特网应用程序，我极力建议要逐渐完全熟悉SQL编码。渐渐熟悉Query Builder也是一种好办法。

3. 词法引用和约束变量

有时，仅用一小段代码将查询参数化就可使查询是可重用的。也就是说，如果能在查询中说明一个替换符，在运行时用一个值替换它，就能在更多的地方使用这个值。

China-pub.com 下载

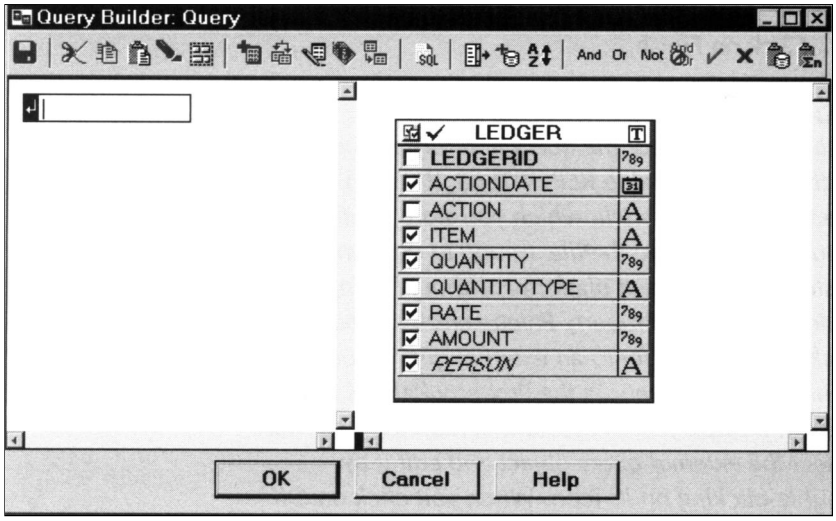


图10-13 Query Builder中的Ledger查询

在Report Builder中有两种方法可以实现这些：可以用词法引用在 SQL语句中替换文本串，或者在SQL语句包含约束变量。词法引用 (lexical reference)就是通过&符号引用报表中的一个列或参数。当引用 SQL语句并且接受Property Palette或运行报表时， Oracle Developer扫描SQL语句，这时它会把变量的值替换成SQL语句中的文本。

约束变量(bind variable)是通过前面加冒号(：)引用报表中的一列或参数，这种结构告诉Report Runtime引擎不是替换成文本而是替换一个值。

注意 引用和变量可以是数据模型列或报表参数，本书建议外部查询只引用参数，如果引用报表的列，会使查询与应用联系太紧密。为了重用外部查询，其他报表必须在数据模型中定义相同的列，这就大大限制了大有好处的重用。关于使用参数的详细内容，参看下一节“参数化应用程序”。你会发现引用数据模型不同层次列的规则是十分混乱的，这倒是往应用程序增加缺陷的一个好办法。

在使用词法引用和约束变量之前，可能觉得它们之间的区别似乎很神秘。实际上在报表中的区别是，可以用词法引用替换整个SQL串，例如替换整个WHERE或GROUP BY子句，但是在需要放置一个字面值如数据值、字符串或日期的场合，只能用约束变量。当在 SQL SELECT或WHERE子句比较中想采用一个值时，通常使用约束变量。在多个报表的基础上，通过语句文本段替换能够设计出一种方法，使得语句可重用性很大，此时通常使用词法替换。在PL/SQL代码中不能使用词法替换，在SQL语句中可以。

比如说有一个报表，想通过一个特殊类型的 Ledger活动，如Bought或 Sold对它参数化，可以用如下的词法替换：

```
  SELECT Person, ActionDate, Item, Quantity, Rate
    FROM Ledger
&WHERE_CLAUSE
ORDER BY Person, ActionDate
```

然而，这种情况通常只用一个约束变量就可以：

```
  SELECT Person, ActionDate, Item, Quantity, Rate
    FROM Ledger
```

China-pub.com
下载


```
    WHERE Action = :ActionType
ORDER BY Person, ActionDate
```

第一个例子采用参数 WHERE_CLAUSE，并且允许替换 WHERE子句的整个文本："WHERE Action = 'Bought'"。尽管格式是特殊和正规的，但对用户来说更重要的是要构造它，而不是要求用户懂得 WHERE子句的语法。第二个例子中，所需要做的就是提供ActionType参数的值：如"Bought"或"Sold"。

不同的报表里要引用不同的表，但 SQL是一致的，这是采用词法引用较好的一种情况。假设可能有几个不同的表，表中有几种不同的人员，在报表中它们共享相同的列，可以用如下的外部查询表示报表查询的SQL：

```
SELECT Name, Age
  FROM &PersonTable
ORDER BY Name
```

参数PersonTable会包含一个确定的表名字，则报表从该表里取得数据。在这个例子中不能用约束变量，因为在FROM子句中没有地方能放一个约束变量，它只能包含用标识符表示的表名字。例如，"FROM 'workingperson'"中单引号会产生一个语法错误，因为约束变量无法去掉引号而把字符串当做一个表名。

使用词法替换和约束变量的规则是十分复杂的，特别是在数据模型中引用不同层次的列的时候。

注意 在从SQL语句引用参数(或列)前，应当先定义并给它们赋初值。在接受 Query Property Palette 时会扫描查询语句并创建变量，但应当采取更多的方法控制这个过程。

10.1.4 命名可视化属性

命名可视化属性(named visual attribute)是一个对象，它在表单或菜单模块中包含一组可视化属性选项设置。这些对象作为所使用的模块对象的格式化代理。当应用命名可视化属性时，对象只采用那些对它有意义的属性外观。当改变属性值时，在各对象中应用该属性的对应值也改变了。表10-2列出了可以采用命名可视化属性设置的各种属性。

表10-2 已命名的可视化属性对象中的属性

属 性	说 明
Font Name	字体的名字(MS Sans Serif, Arial等)
Font Size	字体大小
Font Weight	字体的粗细(Light、Medium、Bold及相互之间的阴影)
Font Style	字体模式(Plain、Italic、Oblique、Underline等)
Font Spacing	字体元素的间隔(Dense、Normal、Expand及相互间的宽度)
Foreground Color	前景区或文本的颜色
Background Color	背景区颜色
Fill Pattern	用背景色和前景色对填充区填充的模式(箭头、小球、胡罗卜、椅子以及其他好的命名方式均可用，但如果想让应用程序是可移植的，则不要使用它们。)
Character Mode	Oracle Terminal资源文件中逻辑属性的名字
Logical Attribute	若应用程序在字符模式或版本下，表示设备属性的基本特征
White on Block	对于单色显示器，如果值为True或Yes，则规定在黑色背景下为白色文本

如果仔细地设计命名可视化属性集，可以将它们从一个模块拷贝到另一个模块来采用标准的可视化属性，而不用对使用可视化属性对象的那些项进行设置。这样，使得只要改变命

China-pub.com 下载

名可视化属性的标准，而不用通过模块去改变所有对象的可视化属性。例如，在第 4章的指导性表单中，想用MS Sans Serif字体代替缺省字体，必须转换所有的文本。可以在初始化文件中改变缺省字体，但另一种办法是在命名可视化属性中，把文本设成 MS San Serif Medium 8。

应当考虑到在应用程序中可能用到的可视化属性的不同组合。为可能用到的每个可视化属性组合创建单独的命名可视化属性对象。将这些对象放进表单或菜单模板中，以便开发者使用合适的属性集来启动。

要在Object Navigator中创建一个命名可视化属性，应找到模块下的 Visual Attribute标题，它就在Report标题下方，点击Create按钮。要在一个对象中应用可视化属性，选择该对象并显示Property Palette，然后把Visual Attribute Name属性设成命名可视化属性的名字。

10.1.5 属性类

命名可视化属性的性质限制了字体、颜色和模式的作用。在表单和菜单块中，能提供更大范围共享属性定义的另一种对象是：属性类。属性类(property class)是属于模块的一个对象，它包含一个属性集。属性可以是任何属性。像命名可视化属性一样，在对象中使用类时，对于所定义的对象有意义的类的所有属性都会起作用。另外，当改变类时，使用该类的所有对象的属性都会改变。

提示 属性类中可以包含命名可视化属性，允许对这两种方法最大程度地重使用。

1. 继承和属性类

属性类为Oracle Developer提供了一种继承(Inheritance)的形式。继承是对象间的一种关系，子对象具有父对象的所有属性。并且可以有任意附加的属性以区别于父对象。在大多数面向对象的程序设计语言中，继承表示了一种层次：孩子是双亲的一个种类。图 10-14表示了面向对象继承层次的一个简单例子。

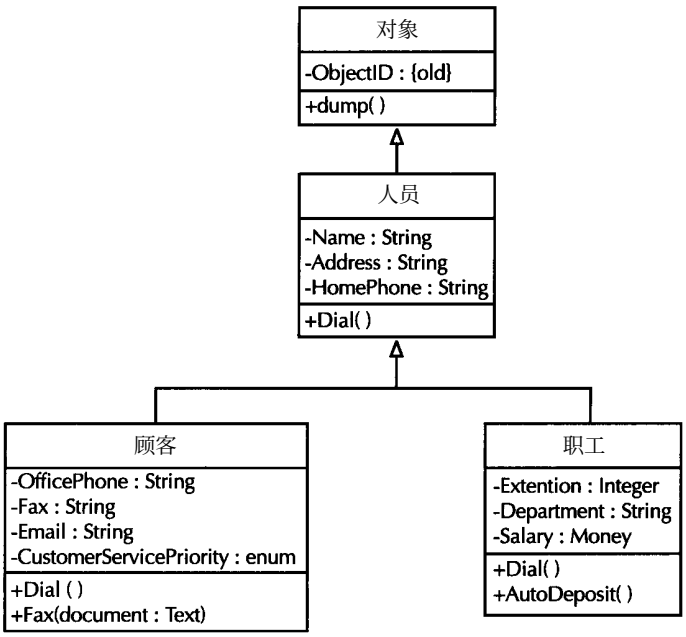


图10-14 继承层次的例子

China-pub.com
下载

根类是Object类，每个对象都有它的子类继承的固定特性，下一级是 Person类，每个Person具有姓名、地址、家庭电话号码以及从 Object继承的属性，再往下一级是 Employee类，每个Employee有范围、部门号、薪水，以及从 Person类和Object类继承的属性。

与Employee在同一层次的是Customer，每个Customer是人员的一种(它继承了 Person类的属性)。Customer具有办公室电话号码、传真号、 e-mail地址及顾客优先顺序。另外，它具有Person和Object的所有属性，所有这些类，决定了类的每个对象的结构，也就是说，可以创建很多不同的 Customer和Employee对象，每个这样的对象都具有与对象相对应类的结构。Object和Person是抽象类，即这样的类没有与其相对应的对象。仅有的目的是表示子类的共享属性。

在Oracle Developer中有属性类和对象间的关系，在某种程度上不同于一般的面向对象方法。在面向对象系统中，当从一个类建立对象时，该对象从类中获得各种属性和方法，通过创建类的附加子类，可以创建具有不同属性的广泛方法的不同种类对象。另一方面，在使用属性类时，当把一个对象与属性类联系在一起时，同时从属性类相对应的属性中获得了属性值，虽然这类似于继承，但实际上是十分不同的。

属性类允许给属性赋缺省值，在大多数面向对象程序设计环境中，还可以为对象增加属性，这并不是属性类的全部用途。属性类支持属性的初始值，需要时可以覆盖它，重新定制对象。对象不能从属性类中取得所有属性，仅仅是那些对象始终支持的值。另一方面，当通过属性类支持触发器时，子类的对象继承了触发器和方法 —— 这是一种关键的重使用方法。因此可以在属性类中一次性编码触发器，在它的子类中重使用。关于在属性类中使用触发器的详细内容，参看“属性类中的触发器”部分内容。

属性类是把可重用性带进模块的一种很好的方法，所需要做的就是抽象属性值并放进共享的父类中，只要在父类中赋值，就同样地能够用于所有的子类，这样就节约了在子类对象中对这些属性编码的时间。更重要的是，当改变父类时，只要父类和子类实际共享了缺省值，并且在对象中没有覆盖它，所有子类会跟着改变。

属性类的另一个优点是属性类可以继承属性类，即允许创建父类的父类，就像在面向对象系统中所看到的抽象类的继承层次，可以使用这种层次结构以得到更精确的属性组合，并且使得属性类重用性更好。例如，可以构造一个巨大的属性类，它包含所有已知的项的属性。当设置好以后，赋值的对象将会从中抽取相关的属性而不管其余的。但是如果在各对象之间有区别，将需要一个不同的相当于块状的 Property Palette，取而代之的是，把不同属性类中间的共享属性抽象成高级的属性类，直到到达最高层，所有对象将共享最高层的属性。最低层类只包含那些与对象有关的属性，这样将会比较容易理解和处理。

注意 给对象划分子类的能力为继承属性提供了一种更通用的方法，详细内容参看本章的“划分子类”一节。通常划分对象的子类，尤其是划分 Object Library对象，主要是为了创建属性类。

2. 创建属性类的分层结构

可以用两个方法中的任一个创建属性类：创建一个新的属性类对象并增加属性，或者用当前对象做为模板，在Property Palette中利用当前对象的所有属性和设置创建一个属性类。

在Object Navigator中Property Palette标题在表单或菜单模块的Program Units正下方，为了创建新的属性类，选择标题并单击 Create工具。

China-pub.com 下载

一旦建立了属性类，双击它的图标显示 Property Palette。在Property Palette顶端的工具条的工具用于控制采用不同方式增加和删除属性 (见图10-15)。

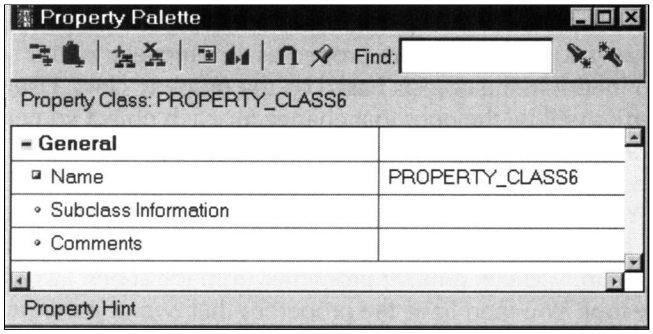


图10-15 Property Palette对话框

单击Property Palette工具条的Add Property工具，显示Properties LOV对话框，从列表中选择一个属性，然后给属性设置想要设的值 (见图10-16)。

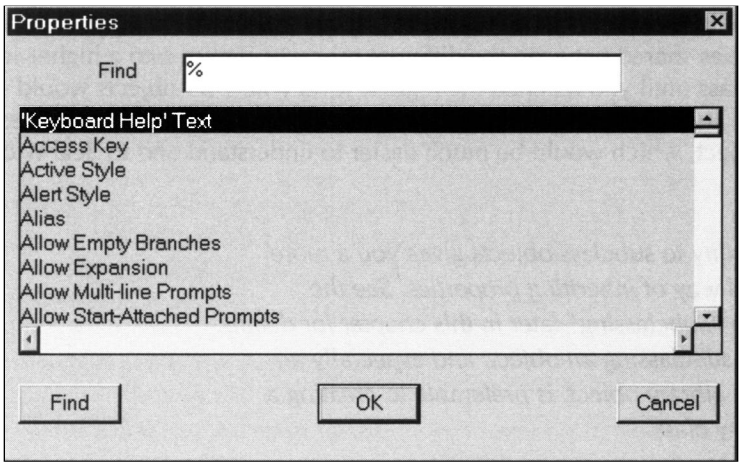


图10-16 Properties对话框

可以从已存在的对象中拷贝属性到 Property Palette中。选择想拷贝对象的所有属性并显示对象的Property Palette，单击Property Palette中工具条上Copy Property工具，拷贝对象的所有属性及设置。

找到并选择属性类对象，单击 Paste Property工具，粘贴所有被拷贝属性的属性类。

要删除不想要的属性，先选择想删除的属性并单击 Delete Property工具。

可以由另一个对象直接创建属性类。首先，选择一个对象并显示它的 Property Palette，单击Property Class工具，通过拷贝所显示对象的属性和值建立一个 Property Palette。Oracle Developer提示正在创建新的属性类并给出了类的名字。然后在 Object Navigator中选择属性类对象并增加属性，或者删除认为不需要从属性类继承的属性。通常，这些删除的属性多是随创建的对象而变化的，例如在画布中的水平位置或垂直位置。

最后一步是创建一个属性类，并从另一个属性类中继承属性。可用前面一样的技术，从一个对象创建一个属性类，只不过这个对象是另一个属性类。如果是从下往上创建类层，使

China-pub.com
下载

用Delete Property工具从超类中删除属性，然后只剩下可供两个或多个属性子类共享的属性。例如，有两个不同种类的文本输入域，一个有特殊的格式掩码并且长度是固定的，另一个有不同的格式掩码并且无固定长度，假定都共享文本输入项的所有其他属性。可以从固定的对象创建FIXEDTEXTFORMAT属性类，然后移去需要为每个对象设置的所有属性，如 X Position、Y Position、Priniary Key等等。对VARIABLETEXTFORMAT做同样的工作。最后从两个类中任何一个创建TEXTFORMAT类，删除掉Fixed Length和Format Mask属性，只留下共享的属性，例如Data Type和Maximum Length。

如果从上往下创建类层，可以使用 Add Property工具及Properties LOV对话框增加属性来创建子类，在这个例子中，应当先创建 TEXTFORMAT类，然后是FIXEDTEXTFORMAT类和VARIABLETEXTFORMAT类。

如果要在多个模块中使用属性类层，象本章前面建议的那样应当把它定义在模板模块中，可以在该模块的基础上建立新的模块，可以用 File | Save As将模板存成一个新模块。也可以从模板中拷贝所需的属性类到新的模块中，可通过拖拉或裁剪粘贴实现。可以把属性类拷贝到模块间可以共享的对象库中，关于划分子类、拷贝及引用对象请参看下面的章节。

3. 从属性类中继承和覆盖属性

对于要从属性类中继承属性的对象，需要把属性类赋给该对象。每个对象有一个叫做Class的属性。为了设置属性类，按通常的方法设置 Class属性，在Object Navigator中选择对象并在Property Palette中选择Class属性。Property Palette提供了一个在模块中属于该对象的属性类名字的下拉列表。

在设置Class属性时，对于对象存在的属性， Oracle Developer把属性类的属性值拷贝到对象的属性中。在对象中这样的属性用一个等号 (=)标记，如果对象的某些属性在属性类中无对应值， Oracle Developer保留它们不变。如果属性类中某些属性在对象中无对应属性， Oracle Developer不拷贝它们。

注意 属性类和划分子类是密切相关的，关于这种关系的内容在“划分子类、拷贝和引用对象”一节中详细讨论。属性类为重使用提供了很多好处，划分子类则提供了更多的好处。

可以用不同的值覆盖对象继承的属性，使得该属性是一个可变的属性。刚把对象的值设好，加号标志就消失了。可变的属性不继承属性类中的变化。

也可以把可变的属性变回成继承属性，选择对象中的属性并且在该对象的 Property Palette中单击工具条的Inherit工具。

4. 属性类中的触发器

属性类一个很好的特点是，不仅可以在属性板中包含有属性，而且在触发器中也可以。这样可以在对象中继承应用程序的功能。在属性类中使用触发器的技巧就是把它们写成可重使用的形式。使用约束变量是有用的，但用处不大，因为引用的名字必须是具有属性类和触发器的表单内的名字，以便触发器和表单结合在一起。定义功能时，除了域本身之外，不需要引用实际的值。这种情况下，这种特点是很有用的。

要创建属性类触发器，在 Object Navigator中部署属性类对象，然后用一般的方法创建一个合适种类的触发器，可以采用 Triggers LOV对话框和PL/SQL编辑器。当将属性类赋给某一个对象时，该对象(表单块或项目)继承了该触发器，并且当相应的事件出现时会运行它，如果

China-pub.com 下载

该事件和与类相关联的这种对象是不合适的，对象则不理会该触发器（例如，把触发器直接放在对象上，对象也不管它）。

可以为对象本身加入相同事件的触发器，从而覆盖继承的触发器。如果部署对象下面的触发器标题，会看到并没有显示继承的触发器名字，只能看到属性类。

10.1.6 划分子类、拷贝和引用对象

Form Builder的Object Navigator提供了一种重用对象的简单方法：在系统中任何地方拷贝和引用对象。例如：至少可以用七种方法实现一个块：

- 从草稿中创建块，如在第4章中演示的那样。
- 通过拖放，从另一个块拷贝块。
- 通过拷贝粘贴，从另一个块拷贝块。
- 通过剪切、粘贴移动块。
- 在模块中按住CTRL键拖拉块。
- 通过拖放块，引用另一个块。
- 通过引用块上的Subclass Information属性，对一个块划分子类。

1. 拖放移动、拷贝和划分子类

当在Object Navigator中选中一个对象并把它拖到其他地方时，可能完成了几种事情之一：拷贝对象、移动对象，或划分子类对象。

- 移动(Moving)对象意味着，在原来的位置把对象删除并连同它的关系，物理地移到拖放或粘贴的位置。可以用鼠标把一个对象简单地拖到一个新位置。
- 拷贝(Copying)对象意味着正创建一个新对象，并用原来对象的大部分内容和关系初始化它。
- 划分子类(Subclassing)对象意味着正创建一个新对象，该对象在一个新位置引用原对象，子类继承了它所引用对象的属性和关系，除了一些可以改变的属性。当改变原对象时，通过继承原对象的变化，引用对象或子类也相应地变化。可以在同一表单或不同表单间划分子类对象，并且可以从对象库中划分子类对象。

拷贝或划分子类对象的最简单方法是，在Object Navigator中从原始位置拖放到新位置。选择对象，如果愿意可以按住CTRL键选择几个类型相同的对象，按住鼠标左键把这些对象拖到适合这种对象的位置。然后会看到一个Forms对话框，提示是拷贝还是划分子类(见图10-17)。

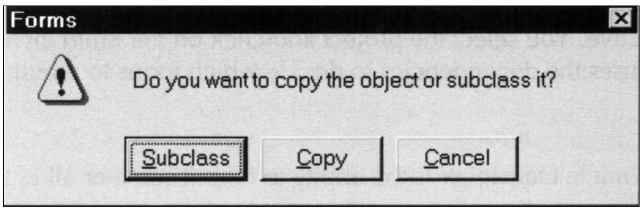


图10-17 Forms对话框

拷贝或划分子类对象都拥有一组对象并且与另一组对象相关联。例如一个块拥有一组项和触发器，并通过它所拥有的项与一组画布相关联。这种关系表现为对象 Property Palette中的各种属性，在 Object Navigator中并不表现成所拷贝对象下面的对象。当拷贝一个对象时，

China-pub.com
下载

Oracle Developer通过Property Palette的有关属性中查找有名字的对象，从而解决模块间的联系。当剪切或拷贝并粘贴一个对象时，会丢失所有联系。

注意 通过引用重用代码，在某一方面增加了系统的耦合度。例如，如果改变了一个模块中的触发代码，而其他模块或应用程序引用了该代码，当该代码有问题时，则所有其他模块通过该代码也继承了这问题。在另一方面，只需在一个地方修改这个问题即可。在修改被引用的代码时要小心。

当在不同模块间建立了引用时，应当用 Project Builder在工程间增加一个关联。如何使用Project Builder中的Add Files to Project工具设置关联，在第4章中“保存表单”已讲过。对于当前这种情况也一样，当在两个模块间有关联时，把源模块的 FMB文件添加到重用模块的FMX文件中。例如，假设有一系列应用程序都应用 Skill模块Skill .FMB中的几个项目，如果在HR表单中实现这些，则在 HR表单和Skill表单之间有一种关联。为了在 Project Builder中表示这种关联，找到HR.FMX文件并单击Add Files to Project工具，然后从Add File对话框中选择Skill .FMB文件。

这种关联的好处并不体现在进行引用的时候，而是体现在修改被引用的原始对象的时候。如果多个模块正在重用该对象，必须重新建立所有重使用该对象的模块以使修改起作用。选择工程并单击Build Incremental工具，则Project Builder应用各种关联决定哪些表单需重建。

2. 划分子类

Oracle Developer划分子类是继承同一类型对象的某些或全部属性的能力。例如，可以把一个文本项建立在另一个画布的基础上，或画布建立在另一个画布的基础上。子类是采取引用方法的对象，父类是被子类引用的对象，子类从父类中继承属性。

Form Builder在为所有表单模块对象设置的General属性中，提供了Subclass Information属性。单击该属性中的More图标，显示Subclass Information对话框(见图10-18)，通过单选按钮选择是对对象划分子类，还是对属性类分子类。

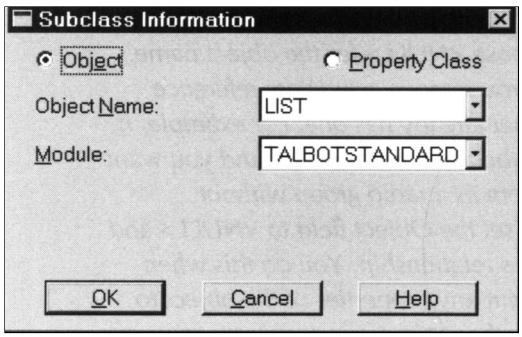


图10-18 Subclass Information对话框

所给出的对话框是特定于块的项的，对于项目对象，还应看到 Block域，对话框中的每个域是一个下拉列表，它从 Form Builder打开的当前模块中获取值。

1) 选择包含要划分子类对象的模块。下拉列表中包含所有打开的表单模块和对象库 (关于对象库详细内容参看“对象库及简易类”部分)。模块可以是包含子类的同一表单。

2) 从所选择模块的块下拉列表中，选择包含要划分子类项目的块。可以选择＜ NULL＞块，这种情况下对象的下拉列表列出模块中的所有项目。也可以选择一个特定的块，此时列表只包含该块中的对象。

注意 不要选择＜NULL＞块。如果有多个对象具有相同名字就会产生问题，刚开始时可以这样做，随着时间的过去，会向应用程序增加更多模块，可能会很容易地增加相同名字的项目，最好使用块名字。

China-pub.com 下载

对象的下拉列表只包含与子类相同类型的那些对象。如果点击列表项目的 Subclass Information属性，只会看到要划分子类的列表项目对象。如果在画布上单击，那只会看到各种画布。

提示 对象的名字可以选择<NULL>，如果一个对象已有子类引用。这种选择可以去掉这种子类引用。例如，如果有一个列表它是另一个列表的子类，现在想把该列表项目重定义为单选按钮组而不用子类引用，可以设置Object域为<NULL>并断开子类关系。当不想让对象从任何其他对象继承任何属性时，可以使用这种方法。

3) 单击OK划分子类对象

在子类中有四种属性：

- 缺省值(Default)：Form Builder从内部缺省值中设置该属性，在 Property Palette中的属性名字旁显示一个小圆圈。
- 设置值(Set)：属性值是用户设置的或是由Form Builder设定的特殊值(例如Layout Wizard可以设置大多数的对象的高度和宽度属性，从而覆盖任何继承的属性值)，Property Palette在属性名旁显示一个矩形。
- 继承值(Inherited)：属性来自于子类对象或父类对象的相同属性， Property Palette在属性名旁显示一个右箭头。
- 覆盖值(Overridden)：属性起初是从父类继承的，现在的值是由用户或 Form Builder赋予的，从而断开了继承性链接。Property Palette上显示一个带红色“×”的继承性箭头，表明断开的链接。

子类只继承在父类或父类的父类中明确设置过的属性。其他任何属性则取 Form Builder的缺省值(一个园圈而非继承性箭头)。对于属性类，可以建立所有父类的整个层次结构，并增加及覆盖住往下各层的属性。

3. 对象库及简易类

Object Library是一个Form Builder对象的容器。对于任意数量的单个对象，如块、项目、画布、窗口或属性类，都可以将其拖进库里让其他模块重使用。

注意 对象库只包含基于表单的对象，不能在对象库中存贮菜单对象、报表对象或图形对象。也不能在对象库中存贮表单模块，而只能存贮表单模块中的对象。

可以用一般方法创建对象库：在Form Builder Object Navigator中选择Object Libraries标题并单击Create工具。该标题夹在 Object Navigator最顶层的PL/SQL Libraries和Build-in Packages之间。如同PL/SQL库一样，对象库对应着一个独立的文件 (如果选择在数据库服务器上存贮就是数据库对象)。当双击New Object Library图标时，Oracle Developer打开Object Library Editor并显示两个空的标签页。在Object Navigator新库的正下方有Library Tabs子标题。所创建的标签页对象就在此子标题下(见图10-19)。

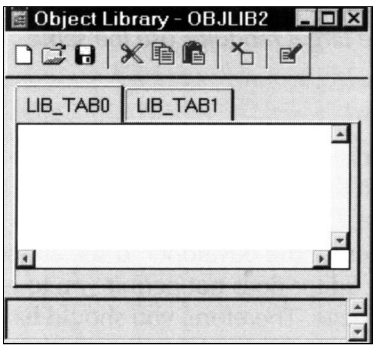


图10-19 标签页

注意 可以用Form Builder主菜单上的Tools | Object Library菜单项或者单击Object Library工具条上的New工具创建对象库。

China-pub.com
下载

组织对象库完全取决于自己。对于要将对象放在所选择的哪个标签页上，没有什么约束。然而，开始之前仔细想一下有助于使库更实用。

首先应当确定建库的目的。例如，对于第 4章的Talbot农场，它们所有应用程序共享一个标准的对象库。这使得项目、画布、窗口以及其他对象在格式和基本属性设置方面是一致的。这样一个标准库包含了一组内容广泛、可重用的对象，它们组织成基于类型的标签页，例如，每个对象类型为一个标签页(块、项目、属性类等各一个)。另一种类型的库可能把块和它的项目包装在一起，或者几个块共享一种主-从关系，以便让几个应用程序重用。对于这种类型的库，可能要想为每一个块创建一个标签页，每页包含一个块及完成该块功能的任何对象。尽管在某些方面也可以用对象组，但类型库非常相似并且可以替换对象组。

提示 在Oracle Developer的演示材料中可以找到标准库。试着在CD-ROM上找具有.olb扩展名的文件，作者曾经看到使用Real Point Coordinate System的库，它不能用于使用Real Inch的Talbot应用程序。必须确信标准库和目标模块采用相同的Coordinate System。

在把对象拖进新库之前还应做另外两个其他的决定。第一，限制每个标签页上对象的数量。数量以在屏幕容易显示为准。根据设置，可以限制数量使得开发者用眼一撇就可容易地看到所有对象。第二，一旦把对象拖到库页上以后， Object Library Editor就不允许重新排列对象。因此，必须想个好办法来安排它们，例如采用字母顺序或一些其他顺序，使得对这些对象是有意义的。

通过Object Navigator及Property Palette可以改变标签页的名字和标记。要创建新的标签页，可在选择当前标签页对象或Tab Page标题之后，使用Object Navigator工具条上的Add工具。可以选择标签页并使用Object Navigator工具条上Remove工具删除。

要将对象包括在库里，只需把它拖到打开的标签页上。也可以在对象上右击鼠标并使用弹出菜单的Add Object to Library菜单项。这种操作把对象拷贝到打开的标签页上，当把对象拷贝到库里以后，用Save工具保存它，可使用Object Navigator工具条上的或者Object Library Editor工具条上的Sare工具。如图10-20所示，打开的标签页上显示的是一系列对象。

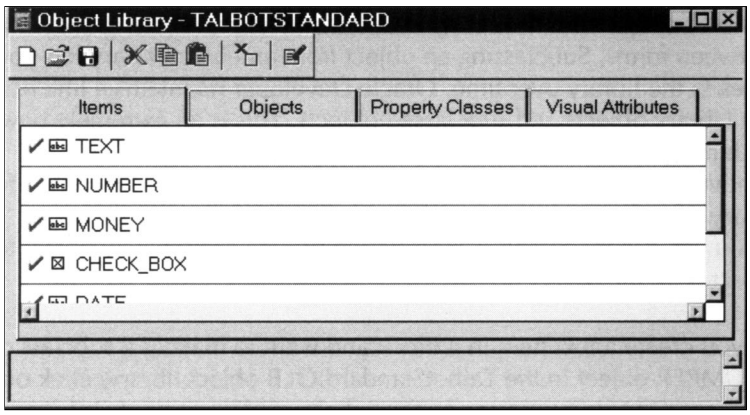


图10-20 标签页上显示的对象

注意 当把对象包括在库里时，实际上是在做这样的事情：拷贝。原始对象和对象库中的对象之间没有链接。如果改变原始对象，在对象库中的对象不会继承这种变化。如果对库中的对象进行改变以改正问题，或增加某种新特征，必须使用Remove Object工

China-pub.com 下载

具删除掉库中的当前对象，并用新的对象替换它，在配置管理时需要特别的小心，因为这种移动会产生应用程序的新版本，目前Oracle Developer没有任何工具可以帮助你。

在Object Library Editor下方的框中显示对象的 Comment属性。在原始对象中用标准Property Palette设置注释，或者用Object Library Editor中的Comment工具在对象库中创建一个注释拷贝。如果要维护独立于对象库的表单中的对象，应在原始表单中加注释而不是在对象库中，因为每次替换库中的对象时必须撤消注释。

要从对象库中重使用对象，则用 File Open工具打开库，然后把对象拖拉进表单中。在表单间拖拉对象时，会出现类似的提示，提示是否拷贝或划分子类对象。从对象库中的对象划分子类允许继承库中有任意变化。 Oracle Developer在Object Library对象和划分子类的对象间维护一种链接。对于需要大量维护和重使用方面，这是一种极为有力的特点。

同样地，可以在表单中创建对象，然后从对象库中划分子类。可用两种方法完成这些。

第一，使用新对象的Subclass Information属性，从对象库中选择一个父对象。

1) 如果在块中创建一个文本项，想使它成为TalbotStandard .OLB对象库中NUMBER对象的一个子类，单击属性上的 More按钮查看Subclass Information对话框(见图10-21)。

2) 将单选按钮设置成Object。

3) 选择TalbotStandard模块(即对象库)。

4) 从库中选择NUMBER对象。

这种方法的缺点是在下拉列表中看到了库中的所有对象，而不是仅仅适合于要划分子类的那种对象。

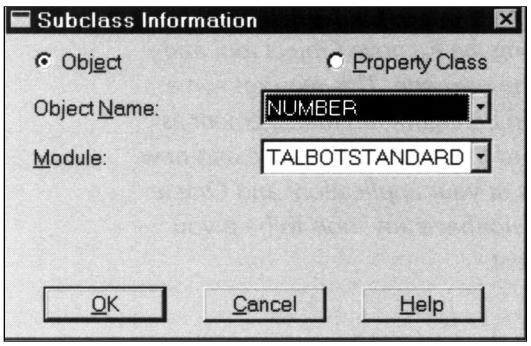


图10-21 Subclass Information对话框

第二，是使用SmartClass特征。SmartClass是库中的一个类，并且可以通过弹出式菜单选择使用。可以把库中的任何对象设成一个 SmartClass。在Object Library Editor中打开对象库并选择想设成SmartClass的对象，从Form Builder的菜单中选择Object |SmartClass菜单项，在Object Library Editor中看到对象的旁边有一个绿色选择标志 (见图10-22)。

在Object Navigator选择一个与 SmartClass相同类型的对象，右键单击对象并选择

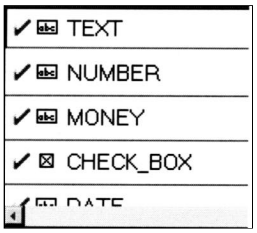


图10-22 对象旁的绿色选择标志

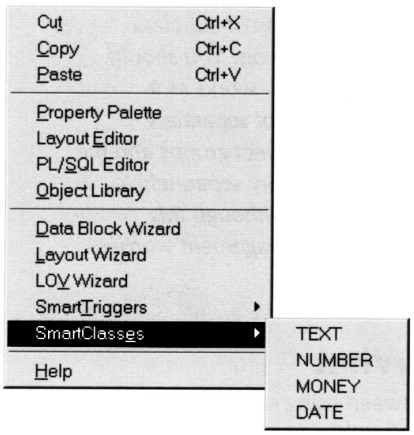


图10-23 对象库中的四个文本项对象

China-pub.com
下载

SmartClasses菜单项。此时会看到一个菜单，它列出了在打开库中的所有与右键单击对象相同类型的SmartClass。这例子显示了对象库中四个文本项对象 —— 通用文本项、数值子类、货币数值子类和日期子类(见图10-23)。

选择SmartClass菜单项，从对象库中的 SmartClass为对象划分子类。这种做法，就像SmartTriggers，提供一种上下文相关的菜单，极大地加快了从对象划分子类的速度。

4. 对象组

Oracle Developer的另一个特点是，允许对可重用的对象打包以准备以后拷贝或划分子类用。对象组(object group)集成了模块单个标题下的一组对象。通过拷贝或引用对象组，可获得它所包含的所有对象。可以对到块级的任何对象进行分组，但不能对块中的项目分组，在对象组中必须包括整个块。

要创建对象组，可在Object Navigator中找到Object Group标题，用Create工具创建新组。然后把所有想集成到组中的对象，通过拖放方法拖到所创建的组的 Object Group Children节点上。

注意 可以用对象库替换Oracle Developer前一版本的对象组，库比组更方便一些，因为库允许访问划分子类的表单中的单个对象。另一方面，可以把对象放进库中做为包的一种，即创建一个对象包。该包中含有在一个应用程序中一次可以拖放或划分子类的对象。当包做为一个系统并且它的一部分不能单独使用时，应当主要使用这种方法。也可以把对象组和组中的对象分别放进库中，同时提供两种重使用方法，但由于重复放置，这会增加配置管理的难度。

10.1.7 什么时候做什么

在划分子类、命名可视化属性及属性类之间怎样选择呢？什么时候应当拷贝对象？什么时候应当引用或划分子类呢？

- 在命名可视化属性中只定义字体、颜色及模式属性，在属性类或子类中可定义任何属性。
- 可以定义属性类或子类，使其分别从属性类或另一个对象继承属性，但命名可视化属性是单独的。
- 可以用PL/SQL动态地改变可视化属性的特性，但不能用这种方法改变属性类的值。在子类中可以动态地继承父对象的变化，但不能改变所有的对象属性，只改变 Oracle Developer允许内部子程序改变值的那些属性，不能动态改变 Object Library双亲属性。
- 命名可视化属性优先于属性类或父对象设置，因此如果两者都有，则字体、颜色和模式将是命名可视化属性的，而不是属性类或父对象的。
- 可以拷贝所需要的任何对象，但不能在一次操作中修改所有拷贝。
- 如果引用一个对象，只能在对象所属的源模块中改变该对象的属性，不能在引用中改变它们。另一方面，如果对某个对象划分子类，则可以覆盖子类中的属性。如果在源模块中有一个错误，则所有的引用或子类都会发生这种错误。
- 对象组使得表单模块可成块使用，如将几个块和触发器作为一个整体。
- 对象库使得Form模块中任何种类的对象是可用的。

从以上的要点可以得出下面的建议：

China-pub.com 下载

- 尽可能用命名可视化属性表示可视化信息 (表10-2列出了可在可视化属性中定义的属性列表)。在属性类、对象库中的对象以及可能做为父对象的对象中包含命名可视化属性，以便以属性形式继承它们。
- 尽可能采用引用父对象尤其是对象库的对象的方法，替换属性类。
- 如果可重用其他模块的主要部分，就使用划分子类的方法，并用对象组创建打包方式的、可重用的模块簇。使用对象库而不要使用对象组，或把对象组放进对象库中以方便引用。
- 拷贝对象只用于进行局部改变的初始设置，也就是说，如果打算在局部模块中定制大量的对象，则拷贝对象而不要引用对象。
- 尽可能使用对象库页来保持对象簇，而不要用对象组。

10.2 参数化应用程序

使用参数可以在运行和编码表单、报表及图形时定制它们。参数 (parameter)是一个变量，当激活对象时，在命令行或调用过程中传递一个称为实参 (argument)的值给它。参数可以构造对外部需求较方便的对象，它允许对象的用户动态地配置对象。

10.2.1 表单参数

表单参数(form parameter)是一个表单变量，可用启动表单时传递的实参给它赋值。向表单传递参数，可通过Forms Runtime程序的命令行，或通过另一个模块中启动表单的各种子程序，如Call_Form、New_Form、Open_Form或Run_Product。每个这样的过程利用一个参数列表作为参数。表单参数可以是下面三种参数类型的任一种：字符型、数值型或日期型。

1. 创建表单参数对象

表单参数是属于表单的对象。要创建它，找到并选择Object Navigator中的Parameters标题，它在Form模块Object Group的正下方，单击 Create工具。双击图标显示参数的 Property Palette(见图10-24)。

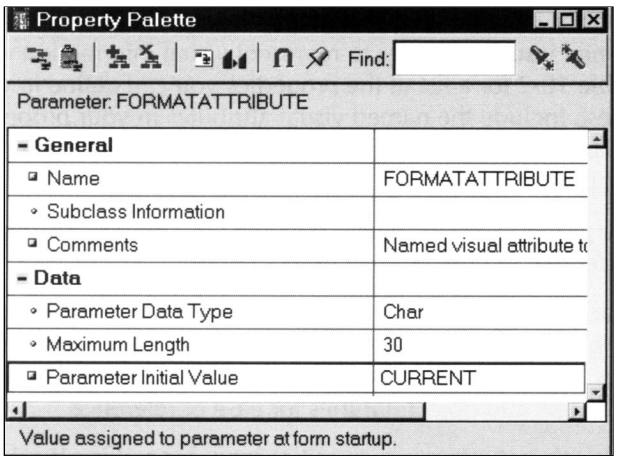


图10-24 参数的Property Palette

这一特定参数接受一个表示命名可视化属性名字的字符串，当启动表单时用于设置表单

China-pub.com
下载

的Current_Record_Attribute属性。这样可以用一组不同的可视化属性中的任一个启动表单。这样做可能是出于可移植性的原因(在不同平台上用不同字体)，或者是为了实现不同可视化标准的不同工程(例如，一个工程是10像素点型，另一个工程是12像素点型)。

需要在属性板中设置参数的Pata Type(字符型、数值型或日期型)、字符型参数的长度及参数的内部名字。当没有给该参数提供实参，而表单要使用它时，还需要设置初始值。不给变量赋初值不好。此例中，缺省值是“ Current ”,表示表单将会使用Current命名可视化属性。

2. 使用表单参数

在PL/SQL代码或特定属性中使用表单参数，可以用约束变量语法。要引用一个参数，在名字前加：“ ：PARAMETER ”，用点分隔符作为前缀。例如，在When-New-Form-Instance触发器中，为了使用参数FormatAttribute设置表单参数，会引用：PARAMETER. FormatAttribute:

```
DECLARE
  vFormID FormModule;
BEGIN
  vFormID := Find_Form(:System.Current_Form);
  Set_Form_Property(vFormID, Current_Record_Attribute,
    :Parameter.FormatAttribute);
END;
```

在表10-3的所列的对象属性中可以引用参数，仍然要使用前缀：“ ：PARAMETER. ”。

表10-3 接受参数的对象属性

对 象	属 性
Block	WHERE子句
Block	ORDER BY子句
List of Values	返回项(在列映射对话框中)
Item	缺省值
Text Item	值下界
Text Item	值上界

3. 传递参数

为了给WorkerSkill表单传递实参，可以把它们放在设置 Runform环境的命令行中：

```
IFRUN60 workersk USERID=talbot/george FORMATATTRIBUTE="WindowsAttrs"
```

其中的USERID是任何表单中都有效的系统参数， FORMATATTRIBUTE是WorkerSkill表单定义的用户定义参数。当Oracle Developer启动并初始化表单时，该参数取WindowsAttrs值。

1) 参数列表 如果想从另一个表单中运行表单，并且需要向表单提供参数值，则在调用嵌入过程运行表单前要先建立一个参数列表。参数列表 (parameter list)是一个包含参数的列表及相应值的对象。通过参数化元素，参数列表让你建立了高度可重用的模块。在通过参数控制的很多不同领域，这样的模块很有用，没有参数的模块是很不方便的。

要创建参数列表，可调用 Create_Parameter_List内部函数。该函数返回 Paramlist型的ID。对于每个想加入到列表中的参数，调用 Add_Parameter嵌入过程。实际列表是创建在表单存储器中的，并永久存在于代码块之外，因此对表单进程只需创建并增加一个列表。要实现这些功能，必须为ID变量取得对象的ID值，测试其值是否为空。在表单会话中，如果代码被调用多次，就要检查列表是否存在，如果已经存在就应当是设置参数而不是增加参数。最后把Paramlist ID传递给调用过程Call_Form、New_Form、Open_Form或Run_Product。在下面的示例中，无名PL/SQL块设置一个参数表，并使用 FormatAttribute参数传递名字“ MotifForm

China-pub.com 下载

VisualAttrs ”。

```
DECLARE
  vListID ParamList :=
    Get_Parameter_list('WorkerSkillFormArguments');
BEGIN
  IF ID_Null(vListID) THEN  -- No list yet, create it
    vListID := Create_Parameter_List('WorkerSkillFormArguments');
    Add_Parameter(vListID, 'FormatAttribute', TEXT_PARAMETER,
                  'MotifFormVisualAttrs');
  ELSE  -- List already created, just set parameter
    Set_Parameter_Attr(vListID, 'FormAttribute', TEXT_PARAMETER,
                  'MotifFormVisualAttrs');
  END IF;
  Open_Form('WorkerSk', ACTIVATE, NO_SESSION, vListID);
END;
```

Oracle Developer初始化表单时自动创建 DEFAULT参数列表。该参数列表包含已经在Form Builder中创建的所有参数。如果使用多重表单，一般要求它们的处理方法是一致的，因此要用参数列表传递参数。可以用如下代码把所有参数直接传给另一表单。

```
DECLARE
  vListID ParamList :=
    Get_Parameter_list('Default');
BEGIN
  Open_Form('WorkerSk', ACTIVATE, NO_SESSION, vListID);
END;
```

2) 传递参数的另一种方法　向表单传递参数的另一种方法是创建一个控制块 (不对应基本表的块)，块中的项目都是参数，并赋于合适的缺省值和其他设置。然后在一特殊的画布上以模式对话窗口(modal dialog window)的形式显示参数表单，并允许用户输入参数值。在应用程序中应当包含一种方法在任何时候可以显示对话框，以便为参数设置不同值。关于创建对话窗口及为窗口创建画布的详细内容参见第 6章。

提示　这些对话框用于可重用对象库中对象是很好的。可以把各参数结构化地构成几个控制块用于表示关系密切的参数簇。然后可以把这些控制块分类成几个不同类型的表单，在每个表单中按需求包括一部分参数。可以把参数入口构造成几个独立选择的对话框，或者合适的话则构成一个对话框。

在内部可以像引用控制块项目一样引用各参数，唯一的区别是不能在命令行上说明参数，并且需在项目前面加控制块的名字而不是加 PARAMETER。

3) 用全局变量向后传递参数　偶尔可能想从被调用表单中传递数据给调用表单。很遗憾，参数表是通过值传递参数，而不是通过引用，因此被调用表单总是处理参数的一个拷贝并且不能直接改变调用表单中参数值。也不像 PL/SQL子程序参数，它没有像 IN OUT或OUT参数一样的使用方法(关于这些参数参看第12章“ 子程序 ”)。

唯一能够找到的解决方法是创建全局变量并在被调用表单中给它赋值。这种方法明显地增加了表单间的耦合度，因为两个表单都引用相同名字的全局变量，这样就无法单独地重使用任何其中一个表单。并且，因为任何地方的代码都可以设置全局变量的值，这样就永远不能真正知道变量的值是什么。因此，必须特别地小心来管理这种变量的值以防止出问题。

在Oracle Developer中有两种类型的全局变量可用。表单全局变量是在任何表单中可以使用并用“ ：Global ”前缀访问的特殊变量。

China-pub.com
下载

程序包变量是PL/SQL包的公共接口规范部分中的变量。任何 PL/SQL代码，包括报表及图形中的代码，通过使用程序包名字引用程序包变量的形式可以访问这样的变量 (如：MyPackage.vVariableName)。关于使用程序包变量的细节参看第 12章。

注意 使用程序包变量而不用表单变量的优点是它们可以跨跃Oracle Developer各组成部分使用。如果在服务器上创建程序包，甚至可以跨应用程序使用。

安全使用全局变量的技巧是在启动应用程序时将它们初始化，然后无论何时使用结束后再初始化它们。这样的话，它们总是反映一个已知的值。例如，假设 WorkSkill表单要调用Skill表单去选择与工人相对应的工作能力，而不是仅仅提供一个工作能力的下拉列表。如果希望用户使用表单的依据例子的查询特征去选择某种工作能力，而这种工作能力是基于它的描述或其他类似的事情的，就应该实现这种方法。为了传递回工作能力的名字，需要一个叫做Skill的全局变量用以存贮名字。在 WorkerSkill表单的When-New-Form-Instance触发器中增加如下语句：

```
:Global.Skill := NULL; —— 将skill初始化为NULL
```

在Skill表单中，可使用任何方法允许用户选择一特定的工作能力，如使用退出时的当前记录，或使用按钮，或使用表示每个工作能力记录的复选框。不管使用什么方法，执行的代码都是给全局变量赋值：

```
:Global.Skill:= :Skill.Skill;    —— 拷贝当前工作能力值到全局变量
```

当返回至WorkerSkill表单时，全局变量反映了 Skill表单设置的工作能力，例如，如果最初是从When-Button-Pressed触发器中调用表单的，那么触发器中在 Call_Form语句之后的代码应当是把全局变量的值拷贝到适当的块项目中，然后把全局变量初始化成 NULL。这样即使触发器执行完并返回到用户以后，也保证总是准确知道全局变量包含的值。

警告 为了重用性及减少复杂性和耦合度，在PL/SQL代码中应避免把参数用做全局变量。也就是说，在从一个程序单元向另一个传递信息时，不要使用设参数值的方法，除非找不到其他方法。可使用PL/SQL程序单元参数移进或移出PL/SQL代码的方法，而不要使用全局变量。

4) 给报表和显示模块传递参数 采用在运行时传递参数来定制表单功能的这种方法，是表单参数提供的一种使表单更易重使用的手段。类似的方法可用于报表及图表显示。大多数情况下，是从表单传递参数给报表和图表。共有两种类型参数 —— 文本参数和数据参数。

文本参数(text parameter)允许给命名的参数传递一个字符串，在被调用的报表或图表中命名参数可以是标准参数、用户定义参数、约束变量或词法引用。标准参数、约束变量或词法引用名字必须对应准确的名字，如 DESTYPE或COPIES。文本参数允许传递任何标准命令行参数给报表或图表。在命令行传递的任何参数一定是文本参数。

数据参数(data parameter)允许传递包含数据记录的 Form记录组的名字。可以传递参数给报表和图形显示，但不能传递给表单。如果传递数据参数给报表，只在主查询中使用，而不能在报表模型的任何子查询中使用。

5) Web. SHOW_DOCUMENT 当把应用程序发布在因特网上时，可使用特殊内部子程序Show_Document链接到一个 Web文本，这包括启动 Oracle Developer应用程序的文档或激活Web报表的文档。例如，为了显示一个静态报表， Talbot事先产生该报表并存贮在 Web服务器

China-pub.com 下载

上的一个逻辑目录/reports中，可以使用触发器中的如下代码用一个超级链接连接到 Web页上：

```
Web.Show_Document('http://www.talbotfarms.com/reports/ledger.html', _BLANK);
```

该调用的第一个参数是一个任意有效的 URL，第二个参数可以是几个可能值之一：

- _BLANK：在一个新的顶层浏览窗口中显示文档。
- _PARENT：在父窗口中显示文档，或者在包含超级连接的框架中显示，点击框架上的超级连接可以显示该文档(类似于_SELF窗口或顶级框架引用)。
- _SELF：在与当前文档相同的窗口或框架中显示该文档。
- _TOP：在与当前文档相同的窗口显示该文档，并替换窗口中当前显示的所有框架使用Show_Document的附加信息，参看第15章。

6) RUN_PRODUCT　运行报表或图表显示时要构造一个参数列表，并用内部子程序Run_Product传递该列表，关于使用 Run_Product的更多内容参见第 15章。下面的代码建立一个参数列表，并运行一个报表，并把 Ledger记录组的数据传递给报表。

```
DECLARE
  vListID ParamList := Get_Parameter_List('ReportArguments');
BEGIN
  IF ID_Null(vListID) THEN  -- No list yet, create it
    vListID := Create_Parameter_List('ReportArguments');
    Add_Parameter(vListID, 'DESTYPE', TEXT_PARAMETER, 'PRINTER');
    Add_Parameter(vListID, 'DESNAME', TEXT_PARAMETER, 'LPT2');
    Add_Parameter(vListID, 'LedgerQuery', DATA_PARAMETER, 'Ledger');
  ELSE
    Set_Parameter_Attr(vListID, 'DESTYPE', TEXT_PARAMETER, 'PRINTER');
    Set_Parameter_Attr(vListID, 'DESNAME', TEXT_PARAMETER, 'LPT2');
    Set_Parameter_Attr(vListID, 'LedgerQuery', DATA_PARAMETER, 'Ledger');
  END IF;
  Run_Product(REPORTS, 'Ledger', SYNCHRONOUS, RUNTIME, FILESYSTEM, vListID);
END;
```

这段代码可以出现在作为菜单项的 PL/SQL块中或在When-Button-Pressed触发器中。当用户选择菜单项或按下按钮时，Oracle Developer使用Run_Product建立该参数列表并传递给报表。这会启动Reports Runtime并与Reports Sever通信，产生所需要的报表。

7) 报表对象及图表项目　尽管Run_Product方法是完全“ OK ”的，但Oracle Developer为图表及报表还在报表对象、图像项目、表单模块的很多种对象之一上提供了非常紧凑的集成方法。报表对象属性包含几个报表系统参数以及一些不同的　Run_Product参数。可以用Property Palette或Report Wizard设置这些参数，然后用 Run_Report_Object内部子程序运行报表，正如第 4章和第 7章演示的。另外，也可以用 Web.Show_Document运行报表(参见“ Web .Show_Document ”部分)。

如图10-25所示例子与以前一样设置 Destination Type和Name属性。不使用数据参数，而是在Data Source Data Block属性中说明用做数据源的数据块。然后报表直接从 Ledger块中取数据，而不需建立一个间接的记录组。这种方法让用户可以更多地控制报表，在　Forms Runtime中用户可以根据例子查询并设置数据块的数据。

提示　在Other Reports Parameters(其他报表参数)属性中可以使用<keyword>=<value>的语法说明任何报表参数，如DECIMAL＝‘ ’。

可以用Chart Wizard或图表项目的Property Palette设置图形在表单中的显示形式。用 Data Source Data Block属性可以设置图形显示的数据源，图表项目增加了 Data Source X Axis 和

China-pub.com
下载

Data Source Y Axis属性，用来说明图表的两维数据所使用的数据块中的项目 (见图10-26)。

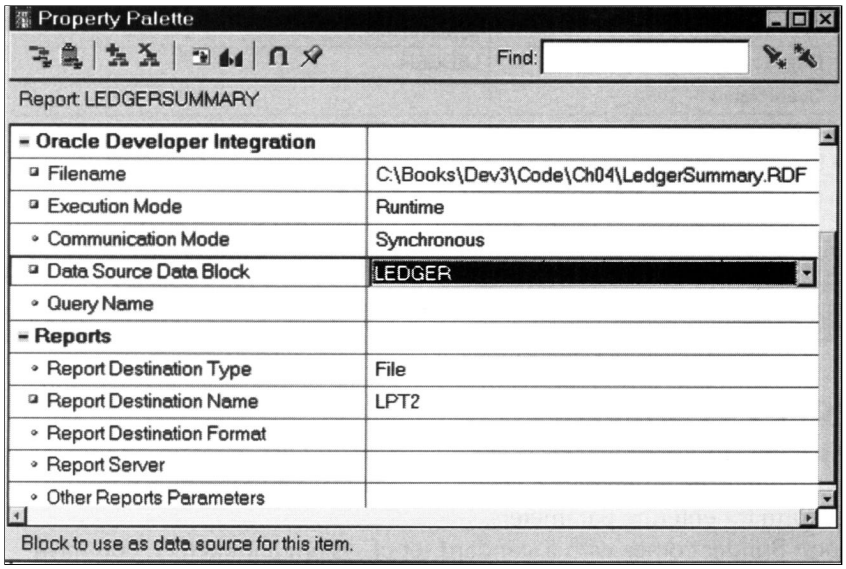


图10-25 用Property Palette设置参数

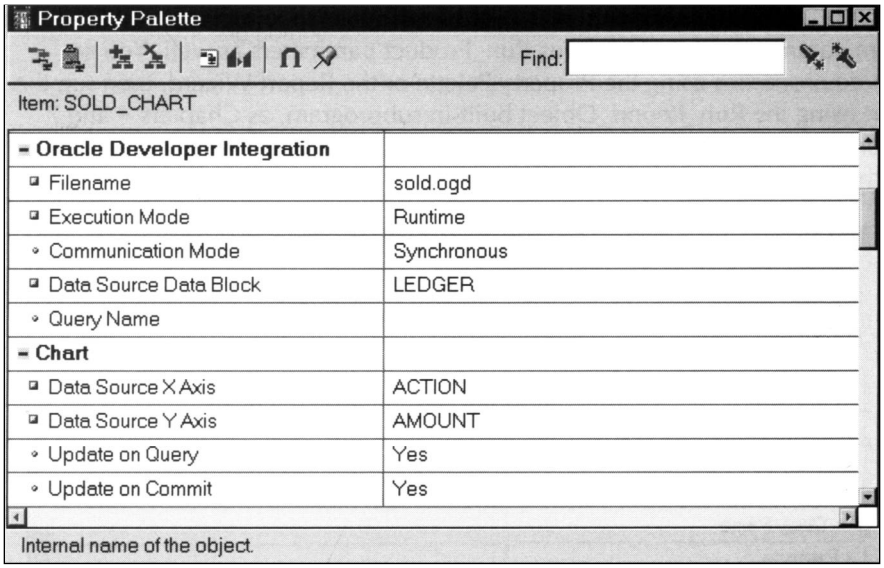


图10-26 图表项目中的两维属性

10.2.2 报表参数

报表参数类似于表单参数，但在使用它们时还有一个附加的选项：在 SQL语句中的词法引用。也可以用嵌入方法创建一个用于输入参数的表单。

Report Wizard有一些标准系统参数，当预览第 4章中的指导报表时，在 Report显示的表单中已经看到了一些表10-4中的参数。

1. 创建参数

China-pub.com 下载

在Object Navigator中创建参数的方法与在Form Builder中的一样。要查找的标题是数据模型标题下的User Parameters(例如Reports | LEDGER |Data Model | User Parameters)。在Parameter Property Palette中，可以输入名字、数据类型、字符型参数的宽度、输入掩码、初始值、有效性验证触发器、注释、值的列表 (LOV)对话框或一组值的组合框。应当输入初始值，并尽量使用掩码、 LOV或组合框，并且/或者尽可能使用有效性验证触发器以确保输入参数是可接受的。

如果在查询中用约束变量引用一个参数可以自动创建参数，则在代码中使用参数时，Report Builder 会自动创建参数。

2. 创建参数表单

在报表对象下面可以发现 Parameter Form对象，它在Layout Object下面。双击该对象的图标会显示一个特殊的 Parameter Form Editor，在其中可为不同参数增加域。如果不建立自己的表单，缺省的表单显示各种系统参数域。当要从草稿中建立一个自己的表单时，它是空白的，要想让用户看到系统参数必须自己建立参数域。可以改变表单的题目，增加提示线文本、样本文本、参数入口域，或把表单分成两页或多页。

较好的方法是用 Tools |Parameter Form Builder菜单项创建一个包含系统参数的参数表单。它会显示Parameter Form Builder对话框，让你选择想显示的系统参数(见图10-27)。

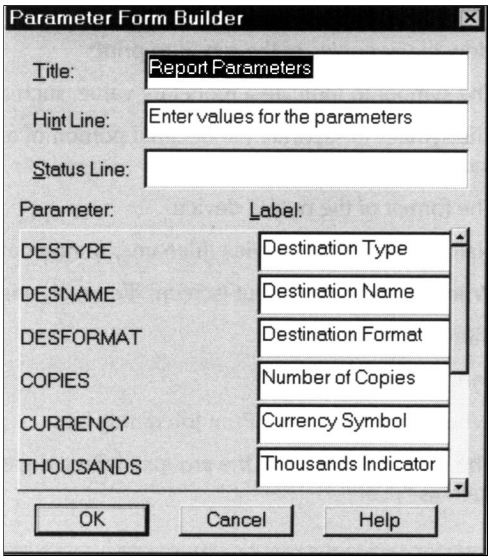


图10-27 Parameter Form Builder对话框

表10-4 报表系统参数

参 数	说 明
Background	是否在后台进程中运行报表
Copies	打印报表的份数
Currency	表示货币值的符号,如“$”
Decimal	分隔数字各部分的符号，如“,”或“.”
Destination Format	输出设备的格式
Destination Name	输出设备的名字(文件名、打印机名字等)
Destination Type	输出数据的地方(屏幕、文件、邮件、打印机、预览器)
Mode	位图或字符形式
Orientation	横向或竖向
Printjob	是否显示Print Job(打印作业)对话框
Thousands	分隔每三位一组数字的符号例如“,”或“.”

要选择一个附加参数或取消已经选中的一个参数，就单击该参数。向下滚动屏幕可看到全部参数，如果愿意，可以键入提示文本或状态条文本。

选择好以后，单击OK，Oracle Developer产生布局并在Parameter Form Editor中显示，然后可以编辑说明文本并加入其他参数(见图10-28)。

在Report Property Palette中有一项Parameter Form Window，用于说明参数表单的页数。也可以说明每页的大小及在Previewer窗口的标题(见图10-29)。

China-pub.com
下载

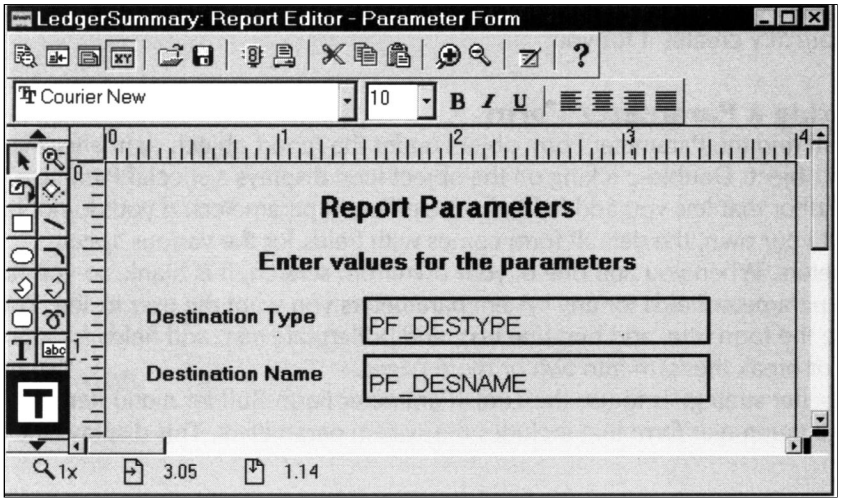


图10-28 Parameter Form Editor产生的布局

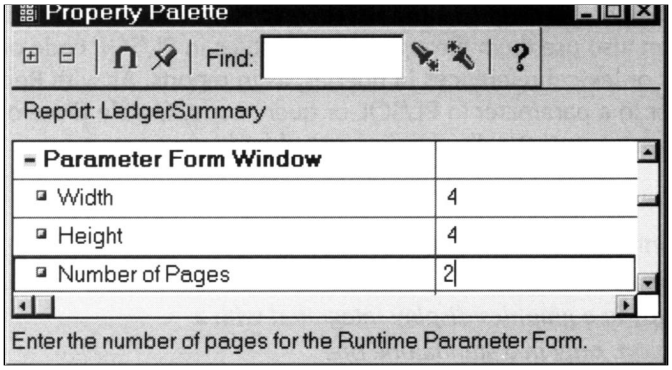


图10-29 Parameter Form Window选项

当增加一个域时，在 Object Navigator或Editor中用通常的方法双击该域就可访问它的Property Palette。如图10-30所示，是PF_DESNAME域非常简单的Property Palette：

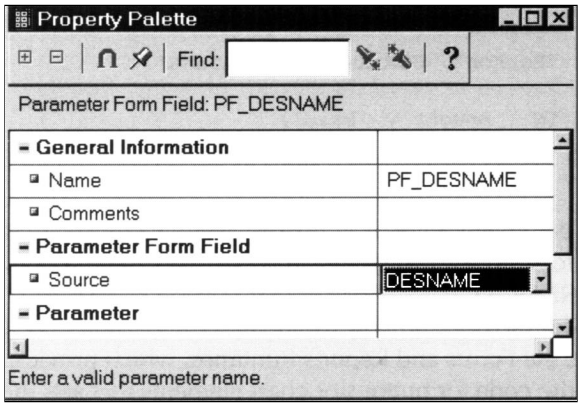


图10-30 PF_DESNAME域的Property Palette

在Source属性的下拉列表中提供了所有参数，包括系统和用户参数。通过设置这些属性，说明参数和域之间的连接关系。

China-pub.com 下载

3. 使用参数

在查询中可以用约束变量或词法引用的形式使用各参数。关于这种方法的细节参看本章的前面“外部查询和Query Builder”部分。可以把参数用做下拉报表设置的一部分，详细内容参看第7章。

10.2.3 图形参数

Graphics Builder的参数与表单参数和报表参数非常相似，但在Graphics Builder中它们扮演着特殊的角色：它们是实现下拉图表功能的基础，并可以传递图表元素的值或图形对象的名字。像在报表中一样，可以通过PL/SQL代码中的约束变量或查询中约束变量及词法引用使用它们。正如在Report Builder中一样，当在PL/SQL或查询代码中引用一个参数时，如果它不存在，Oracle Developer会自动创建一个参数。

注意 关于创建下拉关系图表的详细内容参看第8章。不能创建这样的关系，在图表显示中集成了表单或报表，但它们是相互独立的。

使用Graphics Builder参数的一个区别是，由于在显示模块中没有项目或域，故只能使用参数名，而且没有更多的限制。例如，参数化Talbot购买总量的查询，以便用户可以控制饼图片的顺序：

```
  SELECT Person "Person", SUM(Amount) "Amount"
    FROM Ledger
   WHERE Action IN ('Bought', 'Paid')
GROUP BY Person
&Ordering
```

Ordering参数以词法引用的形式出现，为了按总数排序，把参数设成“ORDERBY2”，为了按人员排序，把参数设成“ORDERBY Person”或“ORDERBY1”。

Graphics Builder中用户无法键入值作为参数，这不像Form或Report Runtimes中可以提供参数输入表单。可以为按钮和图表元素编写代码，设置当用户单击它们时的参数。用户可以通过Graphics Runtime程序的命令行直接输入参数。

10.3 使用多重表单

当建立表单应用程序时，经常面临的选择是建立一个具有很多块的大表单还是一系列的小表单。尽管大表单可以包含更多的功能，但由于它的复杂性，它仅能用于一个小范围的应用程序，因为总会有表单中的某一部分是不需要的。较小的表单，虽然功能较少，但由于它们在一个较小范围内提供功能，可以在较多地方重使用，所以它们的重用性更好。

在构造多重表单的应用程序时，必须面对一系列的问题：

1) 表单间是一种什么关系？

2) 对于事务处理有什么影响？

3) 如何退出不同表单？

以下内容对这些问题给出一个概述。

提示 在过去的几年有人问过几次，可以把多少个表单堆在一起。除了“你的内存允许多少就使用多少”这样的答案以外，对这个问题我从未找到确定的答案。问题大概取决于你用什么样的PL/SQL过程打开表单，详细内容参看下面的“打开表单”。通常，

China-pub.com
下载

最好避免两种情况：在运行时由用户决定打开多少表单或在循环代码中打开表单。应该尽可能保持较少地使用堆栈。

10.3.1 打开表单

在多重表单程序设计中最重要的问题是表单之间的关系。在实际问题中，变成了决定用哪一个内部子程序打开表单的问题。可能选择的方式的范围有：

- 一组相互独立的表单，可以在表单间导航。
- 一组堆栈式的模式表单，只有最顶层才是激活的。
- 单个表单。
- 以上各种情况的组合。

最后两个选择似乎与其他是矛盾的，但实际上不是。多重表单程序设计中的单表单模式表示用一个新表单替换当前表单，在内存中总是只保留一个表单。而各种方式的混合允许替换一个表单而保留其他表单在内存中。

注意 通常在触发器中使用内部子程序打开表单，或在触发器调用的子程序中使用内部子程序打开表单。根据所使用的子程序，Oracle Developer对调用块中的其余代码有不同的处理方式。例如，Call_Form建立一个堆栈，当被调用表单退出时，它返回到调用代码并继续执行。

Open_Form内部子程序从其他所有打开的表单中打开一个独立的表单。调用 Open_Form允许可以访问当前表单，但用被调用表单的菜单替换了当前的主菜单。用鼠标或 Go_Form、Next_Form及Previous_Form内部子程序(此顺序是打开表的次序)可以导航到其他表单中的项目。当进行导航时，除了当前表单的 When-Window-Deactivated触发器及目的表单的 When-Window-Activated和When-Form-Navigate触发器以外，Oracle Developer不做有效性验证，不触发任何触发器。每个表单都有一个当前项目，当导航到本表单的任何其他项目时，通常有效性验证会出现。当说明 ACTIVATE参数时，Open_Form把焦点转移到被调用表单，而不管调用 PL/SQL块中 Open_Form调用后面的任何语句。当说明 NO_ACTIVATE参数时，Open_Form保持焦点在调用表单中并继续执行 Open_Form调用后面的语句。Open_Form不建立保存点(详细内容参看“关闭表单”)。Open_Form带有SESSION参数时，则在新数据库会话中打开表单，NO_SESSION在当前数据库会话中打开表单(详细内容参看下一节“多重表单事务”)。

警告 要非常仔细地使用Open_Form。不能创建这样一种环境：允许终端用户在同一时间内打开过多的表单而使所使用的平台过载。

Call_Form内部子程序通过把当前表单压入一个调用堆栈的方法打开一个表单。当把一个表单压到堆栈时，就禁止了对该表单的使用，同时把焦点转移到被调用表单，使它相对于堆栈中的其他表单变成模式方式。如果说明 DO_REPLACE参数，Oracle Developer用被调用表单的菜单替换主菜单，NO_REPLACE保留当前菜单是可操作的，而不管被调用表单中任何菜单选项设置。如果指定了 HIDE参数，则Oracle Developer隐藏当前表单。而NO_HIDE则保留当前显示。当退出表单时，继续执行调用代码中 Call_Form后面的语句。Call_Form建立保存点并在同一会话中打开一个新表单。也可以指定 QUERY_ONLY参数，则在被调用表单中不允

China-pub.com 下载

许插入、修改和删除。使用Call_Form在任一时刻只有一个表单是激活的。

New_Form内部子程序用新表单替换当前表单。Oracle Developer只从内存中删除当前表单，而保留堆栈中的其他表单或者仍然可以操作的独立表单。Oracle Developer不理会New_Form调用之后的任何PL/SQL语句，因为该表单已不在内存中。New_Form建立一个保存点并在同一会话中打开新表单。还可以指定QUERY_ONLY参数，不允许在新表单中插入、修改及删除。

注意 不管在哪一个表单中，通常应该有相同的主菜单。改变菜单结构容易把用户搞糊涂。例如，用Open_Form及New_Form不能将菜单保留在用NO_REPLACE为Call_Form存放菜单的地方。应当保证所有表单使用相同的菜单模块 (表单属性的Menu Module)。如果需要，可以用Replace_Menu动态地替换菜单。

10.3.2 多重表单事务

回忆第1章，我们知道事务是一个数据库操作序列。当有多个表单打开时，这些表单之间的相互关系对事务有重要影响。第一，可以在多个不同的会话间打开多个表单。第二，使用哪个子程序打开表单，决定了对于当前表单中明显变化的有效性验证及事务处理方法。第三，子程序还决定了当退出被调用表单时发生什么事情 (参看以后章节“关闭表单”)。

如果带SESSION参数使用Open_Form，可以在一个新的数据库会话中打开被调用表单，并因此产生一个不同的事务。缺省是在与调用表单相同的会话中打开新表单。必须把表单运行参数SESSION设成ON或把Registry变量FORMS60_SESSION设成TRUE，才能允许表单的多重会话。用独立的会话打开独立的表单能保持事务是独立的，并可以独立地处理它们，这样降低了表单间的耦合。另一方面，如果表单使用相同的表，当两个事务在同一时刻访问相同的表时会引起加锁冲突。当表单不共享表时独立的会话是很有用的。

在单个事务中，主要问题是数据库相关数据的修改状态。用New_Form或Call_Form要求在调用新表单之前要提交 (commit)或传送(post)数据到数据库中。例如，可能会使用如下的代码：

```
DECLARE
  vCurrentMessageLevel VARCHAR2(2) := :System.Message_Level;
BEGIN
  IF :System.Form_Status = 'CHANGED' THEN
    :System.Message_Level := 5;  -- turn off undesirable messages
    Post;
    :System.Message_Level := vCurrentMessageLevel;
  END IF;

  IF :System.Form_Status <> 'QUERY' THEN
    Call_Form('NEW_FORM', NO_HIDE, NO_REPLACE);
  END IF;
END;
```

这个例子测试是否有任何改变的数据，如果有就传送它们，然后调用一个叫做“NEW_FORM”的表单，并把当前表单放在堆栈中。设置提示信息级别为5来关闭Oracle Developer显示的信息，以防止使用户感到混乱。要记住传送与提交是相对的，传送把数据写到数据库中而不是结束事务。调用Post或Commit_Form的结果是导航到表单级的有效性验证。这意味着在转移到被调用表单前，要验证所有改变并把变化传送到数据库。如果调用一个新

China-pub.com
下载

表单来设置旧表单中的数据，就会产生问题。例如，如果调用一个表单去查询一个复杂查询中的某种ID值，用来设置当前表单所需的某种关系，而在得到设置之前已传送验证了该关系，这就会产生错误。像这种情况，希望使用 Open_Form，但不要求进行传送和有效性验证。

10.3.3 关闭表单

有三种方法关闭表单：

- Close_Form:关闭指定的表单，用这种方法可以关闭应用程序的任何表单，包括当前表单，在这种情况下等同于Exit_Form。
- Exit_Form:关闭当前表单，在缺省菜单Action|Exit中可用，对应Key_Exit触发器。
- Clear_Form:回滚并重置内存中所有表单，在缺省菜单中 Action | Clear All可用，与Key-ClrFrm触发器相对应。

Call_Form和New_Form子程序建立一个Oracle Developer保存点，而Open_Form不建立。保存点(savepoint)是事务中不用回滚整个事务而可以部分回滚事务的一个标记点。当转移到不同表单时，创立一个保存点可以较好地控制关闭被调用表单时所发生的事情。如果有保存点，并且发生回滚事件，Oracle Developer只将修改回滚到调用新表单之前的点，这样保护了已经完成并且已经传送的修改。

Exit_Form和Clear_Form有两个参数 —— Commit模式和Rollback模式。Exit(退出)进程根据Commit模式首先提交所有修改。

这种模式有四种方式：

- ASK_COMMIT:询问用户是否提交修改，然后，如果用户同意 (缺省方式)，就验证修改情况并提交
- DO_COMMIT：不用询问用户，验证修改情况、提交并退出
- NO_COMMIT：验证修改、退出而不用提交
- NO_VALIDATE：不进行验证、提交或询问用户并退出

当用Exit_Form或Clear_Form退出表单时，Commit模式确定有效性验证和提交动作。在提交之后，Rollback模式决定会发生什么事情。Rollback模式有三种可能的取值：

- TO_SAVEPOINT：把所有变化情况回滚到当前表单的保存点，然后退出 (缺省方式)
- FULL_ROLLBACK：回滚到当前事务的开始点，然后退出
- NO_ROLLBACK：不回滚任何修改而退出

New_Form子程序也有Rollback模式，用它可以处理被新表单替代的表单的变化情况。

注意 可以在Key_Exit和Key_ClrForm触发器设置这些选项来重定义这个进程的缺省处理方式。只需把Exit_Form调用放在Key_Exit触发器中，并说明当做标准处理的选项。

很多开发者遇到的问题是Oracle Developer在关闭一个表单时的处理方式。例如，Action | Exit只简单地关闭当前表单。然而大多数应用程序退出时要完成不同的动作 —— 要关闭应用程序而不仅仅是关闭当前表单。如果有多个表单被打开，则 Oracle Developer在关闭最后一个表单之前不会关闭应用程序。可以用一个全局标志改变这种方式。在主程序也就是在第一个显示程序中的When-New-Form-Instance触发器中把该标志初始化成“ FALSE ”。

```
:Global.Exit_Flag := 'FALSE';
```

在调用Exit_Form的代码中，如When_Button_Pressed触发器或Key_Exit触发器中，在调用

China-pub.com 下载

Exit_Form之前把该标志设成“ TRUE ”。

```
:Global.Exit_Flag := 'TRUE';
```

在每个表单的When-Window-Activated触发器中放置如下代码：

```
IF :Global.Exit_Flag = 'TRUE' THEN
  Exit_Form;
END IF;
```

无论何时退出一个表单后会返回到另一个表单，该表单也依次退出，直到关闭最后被打开的那个表单，应用程序也关闭。在标准对象库或表单模板的 When-Window-Activated触发器中，这是一个好的候选办法。

10.4 编写一次代码

在数据库而不是在应用程序中建立过程、函数及触发器，这种可能性在其他章中已简要地提到过，这样做法对于重使用有什么优点呢？

如果正在用Oracle作为数据库服务器，可以选择在应用程序或是在服务器上运行 PL/SQL代码。要做这样的决定，必须考虑两方面问题：代码 /数据的内聚(cohesion)及性能。

怎样把内聚用于触发器及过程呢？正如在第 13章详述的，可以用包和库对过程及函数分组成可重用、内聚的簇。在这种方法中数据库扮演什么角色呢？与标准程序设计语言不同，PL/SQL本质上是一种数据库程序设计语言。不仅能够用它完成纯粹的程序功能，如计算和逻辑判断，也可以用它维护数据库的关系。当有过程或函数簇一起作用于数据库的表时，需要同时考虑数据(表及列)和代码(过程和函数)的内聚。

这种问题在很大程度上类似于数据库的完整性约束机制。因为 Oracle及其他ANSI兼容的数据库管理器支持主键及外键约束，所以在应用程序中不再需要放置维护这种关系的代码。可以定义带有约束的表，并由数据库管理器维护数据。通过在表中放置约束的特征增强系统的内聚。

类似地，当有过程和函数交互作用于特定表及列，完成某一事情时，通常希望把这些过程和函数与表放在服务器上，而不放在应用程序中。因此可以把代码与它所使用的数据放在一起，得以增强内聚。为了让不同应用程序使用，不必在网络上移动代码，任何能访问到数据库的用户都可以访问到过程。有个例子是用过程作为表单数据块的基础。可以把过程放在应用程序中，但因为打算用过程压缩所有对数据的引用，因此应当把过程与数据放在一起，并授予适当的权限来保证访问控制。另一个例子是用过程表示当一个数据库事件发生时一定会发生的事情，如删除某一特殊种类的行或类似的服务器事件。也可以在初始化该事件的每一个应用程序中完成这些，但把代码放进数据库触发器中会更简单并更直接。

正如第6章已经提到的，如果要防止用户通过工具而不是通过应用程序访问数据，触发器及其他数据库过程提供了一种内部访问数据的方法，即通过数据及过程上的权限来控制。触发器保证充分加强约束，防止用户使用修改数据的工具。

还有一些其他相关性能。你会发现，将代码从 Application Server移到数据库服务器会从本质上提高性能并能减少网络流量。这就产生了应用程序分区 (application partitioning)的概念，即把Application Server代码分割成运行在不同服务器上的块，可以提高资源利用率，尤其在网络上更是如此。 Oracle Developer最好的特征之一，就是能够把在应用程序中写的代码拖进数据库服务器中来创建存贮过程。相同语言系统 (PL/SQL及未来的Java)的有效性使这些变为

China-pub.com
下载

可能。

在Object Navigator中通过Database Object标题来访问各个对象，一旦连接了数据库，这个标题显示数据库中定义的所有用户名的列表。每个用户名依次显示属于该用户的存贮 Program Units、Libraries、Tables及Views。

要创建一个存贮过程，在 Object Navigator中选择Database Objects | <USER NAME>|Stored Program Units标题并单击Create工具。则会显示New Program Unit对话框，通过它可以输入过程名和类型(Procedure、Function、Package Spec或Package Body)(见图10-31)。

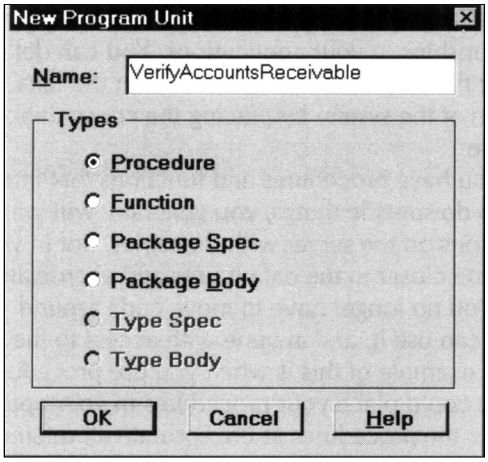


图10-31 New Program Unit对话框

当单击OK时，Oracle Developer显示Stored Program Unit Editor(见图10-32)。

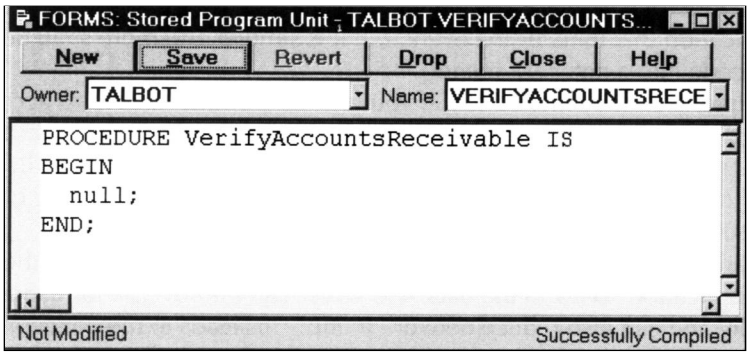


图10-32 Stored Program Unit Editor

然后可以输入程序单元代码并单击 Save按钮编译它。这会导致CREATE PROCEDURE(或其他)语句编译服务器上的程序单元，在运行之后会返回某些 PL/SQL编译错误。对于正创建的这种程序单元，需要具有CREATE权限。

要创建一个特定表的数据库触发器，则部署 Object Navigator中的表对象，会看到在它下面的Trigger标题。和前面一样，单击Create会看到Database Trigger Editor，见图10-33。单击New按钮，建立一个新触发器。Editor中包含几个不同的单选按钮和复选框域来说明触发器的部分内容，如触发器类型 (哪一种语句)及时间(语句前或之后)。在图10-33的例子中，开发者定义Ledger表的Before-Insert触发器。像前面一样，单击Save按钮发出Create Trigger语句编译代码。

注意 在存贮程序单元或数据库触发器代码中不能引用任何表单项。也就是说，不能用约束变量引用数据块中的项目，客户端的项目在执行 PL/SQL的服务器上是不可用的。相反，必须给程序单元的参数增加变量来从程序单元输入/输出信息。这通常也是在模块间进行构程序单元的一种好方法，这样使它们的重用性更好。应该注意到数据库触发器没有参数。

China-pub.com 下载

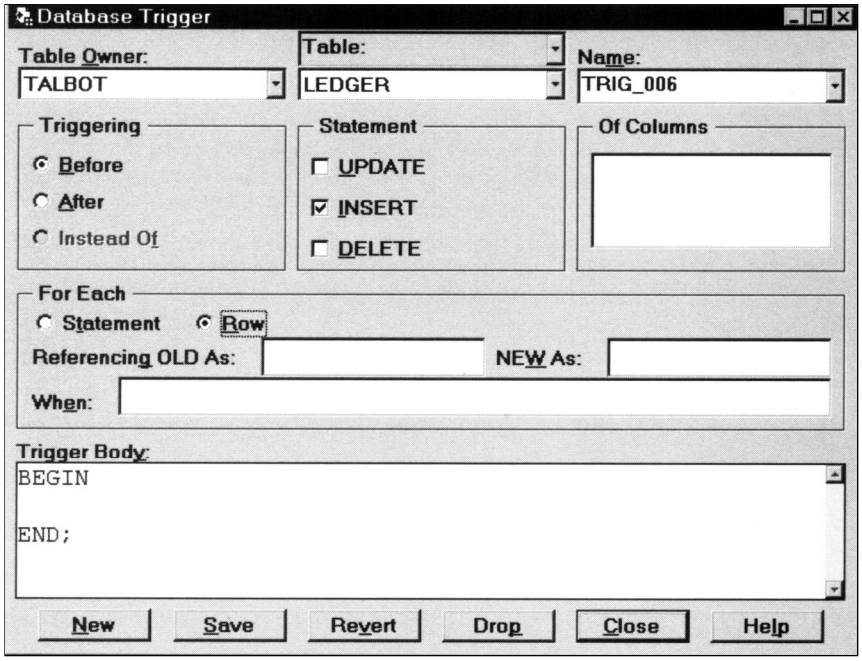


图10-33 Database Trigger Editor

第二部分已经让你领略了原型及其设计的过程，现在需要学会建立可工作的、最终应用程序详细内容的位置，即学会编码、测试及调试技术。如果利用了本书这部分中学到的内容，会发现编码、测试及调试花费的时间会很少，并且很少需要重复工作。

China-pub.com
下载

第三部分　利用Developer编制程序

第11章　PL/SQL基础

本章介绍如何使用 Oracle Developer在原型和设计的基础上建立应用程序。 PL/SQL是Oracle对SQL的过程化扩展。换句话说， PL/SQL使你不仅能执行SQL，也能将其嵌入到控制结构中，如IF-THEN-ELSE和循环结构。与只使用SQL相比，可以使用户在应用中完成更复杂的工作。PL/SQL的语法来自复杂的Ada程序设计语言，但实际上用起来要简单得多。

本章介绍PL/SQL程序设计语言的基础，包括如何组织数据、如何用控制结构编程和如何将SQL作为语言的一部分使用。第 12章将详细讲述使用更复杂的程序结构，这种结构可将程序组织为可重用性代码。第13章介绍Oracle Developer中的调试工具。读完本书的这一部分后，将掌握如何创建、测试和调试应用程序。

本章着重介绍PL/SQL基本编程结构。在Oracle Developer中，可以在两种结构中放置用户代码：trigger(触发器)和program unit(程序单元)。Developer的事件激活触发器，依次调用程序单元。本章讲述如何编写基本触发器和程序单元代码。第 12章介绍如何编制更复杂的程序单元(procedures、function和package)。也可以使用这部分的知识在 Oracle数据库中开发存储程序单元(数据库触发器、过程和包)。第12章的最后一部分讨论如何将程序单元打包为程序库，使不同的应用可以重用这部分代码。

注意　Oracle Developer使用本地PL/SQL版本编译程序单元。Oracle数据库服务器使用当前Oracle数据库服务器PL/SQL的版本编译程序单元。这两个版本的PL/SQL可能不同，所以某些功能在服务器PL/SQL上可以使用而在Oracle Developer PL/SQL上不能使用。例如，在Oracle数据服务器上使用版本为7.3.4，那么在服务器上的PL/SQL版本与Oracle Developer PL/SQL 8相比缺少许多工具。通常，虽然这只对高级用法有影响，如在代码中使用嵌套表，但是需要了解代码的假定，这些假定是说明建立代码的基础是什么。

PL/SQL基本程序单元结构如下：

```
DECLARE
  --Data declarations
BEGIN
  null;--Program statements
EXCEPTION
  --Exception handlers
  WHEN OTHERS THEN
    null;--default handler
END;
```

带有这种结构的完整程序是一个无名块。整个块是一个 PL/SQL语句，因此，在END后需要分号终结符。如果没有数据说明，可以省略 DECLARE、BEGIN和END关键字。如果没有异常处理，可以省略EXCEPTION关键字和WHEN子句。通常包含BEGIN-EXCEPTION-END

China-pub.com 下载

序列结构，而不仅仅只是EXCEPTION子句。在块中至少要包含一个有效的程序语句，因此上面的代码中使用了null(空)语句。在本章的代码例子中，为了使例子能够编译，如果块中没有代码，则其中都包含一个null语句。

注意 PL/SQL块与Oracle Developer Forms的数据块完全不同。PL/SQL块是一段可执行程序(一个无名块、一个过程或一个函数)，而Forms数据块是在一个表单中项定义的集合，这些项可能映射、也可能不映射基准服务器表定义。可以把Forms的数据块看作是记录的列表。在PL/SQL块中可以包含任何级的嵌套PL/SQL块，根据每个块命名的作用域限制所定义嵌套块中的块。PL/SQL是象Ada语言一样的块结构程序设计语言。

下面是应该知道的几个实际问题：

- 大多数PL/SQL表达式使用与SQL表达式一样的语法，可以使用大多数 Oracle SQL内置的函数。如果了解SQL,就了解大部分的PL/SQL。
- PL/SQL不区分大小写字母，所以可以使用大写字母、小写字母，或者为改进可读性而用大小写字母混合编写代码。在引号中的字母 (字符作为其本身使用而不是用作其他东西的符号)区分大小写。
- PL/SQL标识符必须以字母开头，并且只能包含数字、下划线、符号 (#)和美元符号($)。标识符最多为30个字符，并且不能是系统的保留字，如 DECLARE或END。
- 在引号内的字符串中，用两个单引号表示一个单引号，如 'Talbot' s Farm'。
- BOOLEAN型(逻辑型)数据的值为TRUE(真)、FALSE(假)或NULL(空值)，不用引号。例如IS_WINDOWS_DISPLAYED:=TRUE，不使用'TRUE'。
- 可以使用两个横线注释一行或使用一对 /*-*/之间的多行注释：

```
null;     -- This is an inline comment terminated by the line end
/* This is
a multiple-line
comment terminated by */
```

提示 使用行内注释要优于多行注释。因为很容易忘记注释终结符*/或者可能在编写代码时删除了注释对的某一边。

如果大量使用PL/SQL编程，参见Scott Urman的《Oracle 8 PL/SQL程序设计》一书。这本书的参考资料中包含了各种内部函数的完整列表。

11.1 数据概述

关于程序设计语言需要理解的最重要的问题也许就是这种语言如何组织数据。有的程序设计语言有非常复杂的类型系统，而有的却十分简单。有的语言可以扩展类型系统，而有的却不能。

11.1.1 数据类型

数据类型是对数据的分类，如字符、数值或日期。 PL/SQL的程序数据类型与SQL数据类型一致，并扩充了SQL数据类型。由于PL/SQL最主要的目的是嵌入SQL，所以希望直接映射到SQL语句和其中值的类型。 PL/SQL也增加了几种在块程序设计语言中有用的数据类型：BINARY_INTEGER(二进制整数型)、BOOLEAN(逻辑型)、RECORD(记录型)和TABLE(表型)。

China-pub.com
下载

Oracle Developer中使用的PL/SQL增加了几种类型与表单中不同的对象相对应，例如 window(窗口型)和block(块型)。

注意 CURSOR和REF CURSOR是可以与SQL语句联系的特殊数据类型，细节参见11.3节“在PL/SQL中使用SQL”。

1. SQL类型

PL/SQL提供了与SQL一一对应的类型系统，至少是重要 SQL类型是这样。以下这些数据类型的细节参见联机文档： DECIMAL、FLOAT、INTEGER、NUMBER、REAL、SMALLINT、DATE、DATETIME、CHARACTER、LONG、RAW、STRING、VARCHAR和VARCHAR2。在PL/SQL块中可以使用标准Oracle SQL转换函数进行这些类型的显式转换。

2.PL/SQL二进制整数类型和布尔类型

为了处理带符号整型数，PL/SQL增加了三种数据类型。当在PL/SQL中使用带符号整型数进行计算时，不需要将数据像 NUMBER或其他SQL数据类型一样转换为内部格式。在整数运算量很大的块中，使用带符号整数代替 NUMBER数据可以提高性能。

基本的带符号整数类型是 BINARY_INTEGER，可以表示从-2147483647~2147483647范围的整数。两个子类型的范围要小一些。 NATURAL(自然数)类型可以表示从0~2147483647范围的整数，POSITIVE(正数)类型可以表示从1~2147483647范围的整数,可以使用这些数据类型限定非负数值。

BOOLEAN类型可以直接处理TRUE和FALSE。也可以把BOOLEAN数据置为NULL，意思是其值没有定义。这意味着PL/SQL Boolean值有三种可能的状态，TRUE、FALSE和NULL，其逻辑是三值逻辑。

注意 三值逻辑可以是反直观的(counterintuitive)。例如，比较两个Boolean变量，两个中的一个或两个都为NULL，结果为NULL，而不是TRUE。如果在IF语句中表达式值为NULL，则语句执行ELSE子句。第三个例子：NOT(NULL)等于NULL。关于三值逻辑及其在SQL中的使用的讨论参见Chris Date 的《Relational Database: Selected Writings》(Addison—Wesly，1986)第15章中的“Null Values in Database Management”(在数据库管理中的空值)部分。数据库管理器不是一贯地采用三值逻辑的一种模型，所以要仔细查阅Oracle文档，理解在特定情况下会发生什么。

3. PL/SQL的记录

用PL/SQL记录可以定义包含几个数据元素的单一变量，很像数据库中的行。可以使用RECORD类型创建任何类型的结构化数据。然后可以使用圆点符号像逻辑单元一样引用数据。

创建记录，必须首先创建一个独立的记录类型。例如，创建一个对应于部分 Ledger表的类型，可以在DECLARE(声明)部分采用如下语句：

```
DECLARE
  TYPE LedgerType IS RECORD (
    LedgerID VARCHAR(25) NOT NULL := 0,
    ActionDate DATE,
    Action VARCHAR(8),
    Amount NUMBER(9,2));
BEGIN
  null;
END;
```

China-pub.com 下载

在这里，记录只描述表的一部分数据。NOT NULL(非空)子句与SQL中NOT NULL一样，阻止用户给域赋值NULL。尽管RECORD类型常见的应用是描述与表数据相对应的数据，但是在其中可以包含任意类型的域，包括RECORD类型。可以在程序包中使用记录。例如描述未存储在数据库中的运行时对象，如一天24小时的特定时间。可以用对记录中数据进行操作的一组函数封装类型，那样，就可以代入时间值和时间间隔的值去调用Subtract函数求出它们的差值。

```
PACKAGE Time24Package IS
  TYPE Time24Type IS RECORD (
    hour NATURAL NOT NULL := 0,
    minute NATURAL NOT NULL := 0,
    second NUMBER NOT NULL := 0);

FUNCTION Subtract(time Time24Type,
                  interval TimeIntervalPackage.tTimeInterval)
    RETURN Time24Type;
END;
```

可以说明两个Time24Type类型的变量，把它们传递给函数，取回差值，把它赋给第三个Time24Type类型的变量。这种封装和抽象数据类型可以描述比简单PL/SQL类型更复杂的对象。第12章给出了关于封装和抽象数据类型的完整的讨论。

注意 参见11.1.3节“使用类型属性”，该节介绍了使域类型与数据库列类型一致的方法。

一旦定义了类型，就可以使用类型创建变量，在DECLARE部分也一样：

```
DECLARE
  TYPE tLedger IS RECORD (
    vLedgerID VARCHAR(25) NOT NULL := 0,
    vActionDate DATE,
    vAction VARCHAR(8),
    vAmount NUMBER(9,2));
  vLedgerItem tLedger;
BEGIN
  null;
END;
```

这个记录包含一个值为0的LedgerID域和其余值为NULL的域。PL/SQL在执行DECLARE部分时创建变量，在执行定义变量的块的END时释放变量。如下面代码段中所示，可以给域赋值，使用圆点符号引用域：

```
IF vLedgerItem.vAction = 'Sold'
THEN vLedgerItem.vAmount := 0.0;
END IF;
```

注意，比较操作符为“ = ”，然而赋值操作符是“ := ”。

4. PL/SQL表

尽管RECORD类型功能强大，但是缺乏一个功能：在同一时刻描述多于一行的数据的能力。这就是TABLE类型的工作。大多数程序设计语言都有这种多值 (multiple-value)数据类型,PL/SQL也不例外。

Array(数组)在程序设计语言中是表示多值的变量，其值标为元素可以通过一个整数下标访问它。数组第一个元素的下标是1，数组第二个元素的下标是2，以此类推。有的语言的数

China-pub.com
下载

组下标从0开始，而不是从1开始，还有一些是任意值，包括负数下标。PL/SQL的表像数组一样通过下标访问。与大多数程序设计语言的数组类型不同的是，用户完全控制下标和分配给元素的存储器。在给使用某个下标的元素赋值时，如果元素不存在，PL/SQL则创建它。因此可以建立一个包含五个任意下标值(如-4、33、1275、2和-43)并且有五个元素的数组。访问一个未用这种方法创建的元素，将得到Oracle错误代码-1403——“没找到数据”。

Linked list(连接列表)在程序设计语言中是描述通过指针(first指向第一个、last指向最后一个、next指向下一个或previous指向前一个)访问元素的数据结构。PL/SQL表在存储上像一些链接列表，只有加入到表中的元素才存在。与链接列表相似，可以使用表属性导航表。与链接列表不同的是，如果知道一个元素的下标，可以直接找到这个元素。

PL/SQL的表将数组和连接列表的最好的特点结合起来。可以使用表来达到许多不同的目的。主要的是在PL/SQL子程序中描述多值数据和在PL/SQL代码中描述查询返回的结果集合。例如表使用户在存储过程中建立一个完全的结果集合，并且通过IN OUT参数返回给调用者。

注意 Oracle Developer Forms有一个类似的结构叫做record group(记录组)。然而这些对象只在一个表单中存在，因为它们不是PL/SQL语言不可或缺的一部分，必须使用内部子程序访问数据元素。在通用PL/SQL编程中，使用表代替记录组。但有些功能需要使用记录组，如建立动态列表(第7章)或传递Oracle Developer参数列表(第10章)。

1) 声明表　和记录一样，用声明表类型来声明表，然后声明该种表类型的变量。这使表对于打包程序设计格外合适(参见第12章)。可以通过程序包的说明导出一个表类型，然后在引用程序包的PL/SQL代码中声明满足所需数目的这种类型的表。

可以声明一个简单的数值表：

```
DECLARE
  TYPE tLedgerAction IS TABLE OF Ledger.Action%TYPE INDEX BY BINARY_INTEGER;
  vActionTable tLedgerAction;  --declare an action table
BEGIN
  null;
END;
```

在这种格式中需要INDEX子句，因为PL/SQL除了BINARY_INTEGER索引以外还不支持其他的索引。也可以声明一个更复杂的使用记录类型的表：

```
DECLARE
  TYPE tLedger IS TABLE OF Ledger%ROWTYPE INDEX BY BINARY_INTEGER;
  vLedgerTable tLedger;  --declare a Ledger table
BEGIN
  null;
END;
```

可以在TYPE声明的OF子句中使用任何记录类型，不仅是%ROWTYPE派生记录。

2) 用下标和属性使用表　一旦声明了表类型和这种类型的变量，就可以通过给它们赋值建立表中的元素。在使用下标给元素赋值之前，PL/SQL并不真正建立任何元素。

例如，通过直接给vLedgerTable变量赋4种总帐动作(action)的方法，建立一个可能值的列表。

```
DECLARE
  TYPE tLedgerAction IS TABLE OF Ledger.Action%TYPE INDEX BY BINARY_INTEGER;
  vActionTable tLedgerAction;  --declare an action table
BEGIN
  vActionTable(1) := 'Bought';
  vActionTable(2) := 'Sold';
  vActionTable(3) := 'Paid';
  vActionTable(4) := 'Received';
```

China-pub.com 下载

```
  -- Use the table in further processing
END;
```

在表名后面的括号中指定下标，引用表中特定元素。在这个例子中，通过下标 1、2、3、4在表中建立了四个元素。

注意 可以使用任何下标值，不只是顺序的或从1开始的序列。然而，为了便于记忆和在循环中编程，最好使用这些序列。可以使用SQL来填充表(参见11.3节“在PL/SQL中使用SQL”中关于CURSOR和循环语法的细节)：

```
DECLARE
  TYPE tLedgerAction IS TABLE OF Ledger.Action%TYPE INDEX BY BINARY_INTEGER;
  vActionTable tLedgerAction;  --declare an action table
  CURSOR cLedger IS SELECT DISTINCT Action FROM Ledger;
BEGIN
  FOR vLedgerAction IN cLedger LOOP
    vActionTable(cLedger%ROWCOUNT) := vLedgerAction.Action;
  END LOOP;
END;
```

这个例子从Ledger表中通过一个cursor和特定cursor FOR loop选定Action列的DISTINCT(不同的)值，用特殊的cursor ROWCOUNT属性做为下标把检索值赋给动作(action)表。这和前面的例子的结果相同，但是更灵活。如果表中值的可能集合改变了，第一个例子必须重做，而第二个不用。

也可以通过表11-1所列出的表属性访问表。

表11-1 PL/SQL表属性

属 性	类 型	描 述
Count	NUMBER	返回表中元素的个数
Delete		删除表中所有的元素
Delete(index)		删除表中index指定的元素
Delete(index1,index2)		删除表中所有在index1和index2之间的元素(包括index1和index2)
Exists(index)	BOOLEAN	返回在表中是否存在index指定的元素
First	BINARY_INTEGER	返回表中第一个元素的下标
Last	BINARY_INTEGER	返回表中最后一个元素的下标
Next(index)	BINARY_INTEGER	返回表中下标指定的元素的下一个元素的下标
Prior(index)	BINARY_INTEGER	返回表中下标指定的元素的前一个元素的下标

将表名和圆点符加上表属性来使用表属性。例如，为取得表中元素 4的前一个元素的下标，使用以下语法：

```
vIndex := vActionTable.Prior(4);
```

使用这些属性的例子，参见《PL/SQL User’s Guide and Reference》或Scott Urman的书《Oracle 8 PL/ SQL程序设计》。

5. Oracle Developer的PL/SQL对象类型

在Oracle Developer中使用PL/SQL时,有几种描述不同对象的类型。表 11-2列出了Forms中不同的对象。

表11-2 Form PL/SQL对象类型和查找函数

对 象	类 型	查 找 函 数
Alert	ALERT	FIND_ALERT
Block	BLOCK	FIND_BLOCK

China-pub.com
下载

(续)

对　象	类　型	查找函数
Canvas	CANVAS	FIND_CANVAS
Editor	EDITOR	FIND_EDITOR
Form	FORMMODULE	FIND_FORM
Item	ITEM	FIND_ITEM
List of Values	LOV	FIND_LOV
Menu item	MENUITEM	FIND_MENUITEM
Parameter List	PARAMLIST	GET_PARAMETER_LIST
Record Group	RECORDGROUP	FIND_GROUP
Record Group Column	GROUPCOLUMN	FIND_COLUMN
Relation	RELATION	FIND_RELATION
Report	REPORT	FIND_REPORT
Tab Page	TAB_PAGE	FIND_TAB_PAGE
Timer	TIMER	FIND_TIMER
View	VIEWPORT	FIND_VIEW
Window	WINDOW	FIND_WINDOW

在调用带参数内部子程序时，通常要提供对象的唯一名称或对象 ID。对象ID是唯一识别对象的对象句柄。当使用表 11-1中的类型声明一个变量时，实际上是声明了一个对象 ID句柄。通过调用FIND函数，可取得对象 ID，FIND函数在你传入对象名时返回 ID。因此可以用对象名得到对象标识符 ID，然后在一系列子程序中使用 ID。如果多处引用对象时，这可以加快处理的速度，因为每次传递对象名时， Oracle Developer内部都要查找其ID。例如，在循环处理中有一段涉及item(项)值的代码。应用FIND_ITEM()函数在进入循环之前取得 item的ID，然后在循环中使用item ID来查阅item 的值，而不是在每次循环中通过 item名字来查阅 item。

参数列表对象有一个特殊的 ID类型PARAM_LIST，对应GET_PARAMETER_LIST函数而不是用FIND开始的函数。

表11-3列出了Graphics数据类型和其对应的 GET函数，尽管名称约定不同，但与 Forms类型的工作方式基本一样。

表11-3　Graphics对象类型和查找函数

对　象	类　型	查找函数
Axis	OG_AXIS	OG_GET_AXIS
Button Procedure	OG_BUTTONPROC	OG_GET_BUTTONPROC
Chart Template	OG_TEMPLATE	OG_GET_TEMPLATE
Display	OG_DISPLAY	OG_GET_DISPLAY
Field Template	OG_FTEMP	OG_GET_FTEMP
Graphic Object	OG_OBJECT	OG_GET_OBJECT
Layer	OG_LAYER	OG_GET_LAYER
Query	OG_QUERY	OG_GET_QUERY
Reference Line	OG_REFLINE	OG_GET_REFLINE
Sound	OG_SOUND	OG_GET_SOUND
Timer	OG_TIMER	OG_GET_TIMER
Window	OG_WINDOW	OG_GET_WINDOW

注意　报表没有对象类型和对象ID。

China-pub.com 下载

11.1.2 声明变量和常量

在本书的前面部分中已经看到了几种变量声明。变量声明按如下形式在 DECLARE部分中声明：

```
variable_name  type_name  [NOT NULL] [:= initial_value];
```

也可以在子程序(过程或函数)的参数列表中声明变量。在子程序 (过程或函数)的参数列表中声明变量，参见第13章。

NOT NULL限定词说明变量的值不能为NULL(空值)。初始值的指定给指定为文字或表达式的值设置变量。如果指定 NOT NULL，则必须指定初始值。初始化所有的变量是一个很好的方法，这样可以确保用户标识符在第一次被引用时有一个有效的值。也可以在赋值时使用保留字DEFAULT，用:DEFAULT initial_value代替:=initial_value。

可以通过宿主变量引用Oracle Developer对象。宿主变量是嵌入应用的一个变量名，通过在变量名前加上冒号 (:)来引用它。可以在使用 PL/SQL变量的任何地方使用宿主变量，包括SQL语句内和语句外。

```
BEGIN
  /* Update the age in the block field and the database. */
  :Person.Age := :Person.Age+1;
  UPDATE Person SET Age = Age+1 WHERE Name = :Person.Name;
END;
```

PL/SQL和所有的Oracle Developer产品的类型一样，所以宿主变量在 PL/SQL过程中不需要单独的声明。PL/SQL把它们看作是已经声明的在item定义中指定的类型。

注意 Oracle Developer对象使用VARCHAR2类型表示文本值。CHAR、VARCHAR、VARCHAR2之间的不同在特定的情况下是十分重要的。CHAR描述定长度数据。如果给一个CHAR变量赋一个比规定值的长度要长的值，PL/SQL就会填入空白字符串，其长度是变量的长度。VARCHAR和VARCHAR2一样都不用这种方式填充。在赋值和比较时可能会有问题。如果给CHAR变量赋一个比其长度要短的值，PL/SQL会填充这个字符串。如果在VARCHAR2变量中有末尾空格，当它与其他任意一种类型的字符串比较时，PL/SQL会把它们看作是比较的一部分，但是并不填充其他字符串。如果比较两个CHAR变量，PL/SQL在比较之前会把比较短的一个填充到与比较长的一样长。这意味着如果在比较长的字符串中有末尾空格，PL/SQL实际上将通过为比较短的字符串填充空格忽略它们。一般来说，在大多数情况下，应该使用 VARCHAR2，避免使用CHAR和VARCHAR。

常数是在声明时赋值并且以后不能再赋值的用户标识符。用 CONSTANT关键字声明常量：

```
constant_name   CONSTANT   type_name  := value;
```

PL/SQL要求初始化常量，常量的初始值是常量整个生存期的值。

11.1.3 使用类型属性

PL/SQL类型属性是一个修饰符，用来取得一个对象在声明其他对象时信息。% TYPE属性可以取得变量、常量或数据库列的类型。% ROWTYPE属性可以取得数据库表或 SELECT Cursor结果表的所有列的类型。尽可能地使用% TYPE属性，这可以大大地提高程序的可

China-pub.com
下载

维护性。

可以使用%TYPE说明与另外的变量或常量同一类型的变量。也可以用它来说明与数据库列类型相同的变量，因为通常要在变量中存储与列类型一样的值。

下面的例子代码从数据库中检索一个值，增加这个值，并把结果赋给一个与其类型一样的新变量。

```
DECLARE
  vDBAge      Person.Age%TYPE;    -- database column type
  vNewAge     vDBAge%TYPE;         -- type from dbAge
BEGIN
  SELECT Age INTO vDBAge FROM Person
   WHERE Person.Name = :Person.Name;
  vNewAge := vDBAge + 1;
END;
```

注意 即使利用NOT NULL属性声明一个数据库列，PL/SQL也不会给变量加上NOT NULL限定词。如果要防止给变量赋予NULL，必须明确地说明。

可以使用%ROWTYPE说明能够创建一个包含表所有列的记录变量。例如，在一个触发器中为了进一步的处理，要从数据库中检索 skills(技能)和abilities(能力)。下面的声明创建了一个保存WorkerHasSkill表所有列的记录。

```
DECLARE
  hasSkillRecord   WorkerHasSkill%ROWTYPE;
BEGIN
    -- Retrieve records and process; for example,
hasSkillRecord.Name := 'Gerhardt Kentgen';
END;
```

%ROWTYPE属性也可以应用于对cursor SELECT结果产生的表。使用%ROWTYPE，可以创建一个PL/SQL自动正确格式化的、能够存放 SELECT返回的任何数据的记录。如果使用%ROWTYPE，不需要声明每个变量的类型，使维护更简单。例如，要对 Ledger表中按person分组求和的amount的进行检索，可以按如下方式说明变量：

```
DECLARE
  CURSOR sumCursor IS SELECT Person, SUM(Amount) Total
      FROM Ledger GROUP BY Person;
  sumRecord   sumCursor%ROWTYPE;
BEGIN
   null;  -- Retrieve rows and process
END;
```

当按这种方式使用%ROWTYPE时，必须为SELECT列表中的任一个表达式指定一个别名，比如在sumCursor SELECT中的TOTAL一样。这个别名在PL/SQL创建的记录中成为字段的变量名。关于cursor(游标)的更多细节，参见11.3节“在PL/SQL中使用SQL”。

如果两个%ROWTYPE变量来自同一个table或cursor，可以把一个变量赋给另外一个变量。不能把用%ROWTYPE创建的记录赋给用 RECORD类型创建的变量。除了使用 cursor以外,RECORD类型与%ROWTYPE声明的使用方式基本相同。RECORD的一个缺点是不能从数据字典自动创建其结构。优点是可以为其指定任意的结构，包括不对应数据库列的附加字段。

11.2 程序控制简介

程序设计语言的控制语句是程序设计语言的主要特点，PL/SQL尤其是这样。控制结构将

China-pub.com 下载

面向过程程序设计语言区别于面向说明程序设计语言（如SQL)。在PL/SQL中嵌入SQL，结合了数据说明语言和控制结构及SQL语句的优点。

控制结构有三种：顺序结构、选择结构和循环结构。

顺序结构是程序语句的简单顺序。PL/SQL按顺序执行顺序结构，每次一条。SQL工作在语句接语句(statement-by-statement)的基础上，SQL语言每次只理解一条单个的语句。PL/SQL处理一系列语句，并且有变量连接到语句。这为结构化处理提供了更大的灵活性，用户可以把处理分割为多个步骤，包括复合的SQL语句。

选择结构是根据真或假的逻辑条件决定程序分支的语句。使用选择结构为进一步处理做判断，这在早期的SQL中是无法实现的。尽管DECODE函数提供了在行接行(row-by-row)基础上有限的判断能力，但是数据说明语法的复杂条件很快就超过了其能力。

迭代结构是控制重复执行的循环过程的语句，SQL作为内部机制，为循环提供了几种途径。除了对结果表的多行的循环外，SQL嵌套了子查询，可以在高层结构中实现循环，这是使用谓词演算操作符(例如ANY和ALL，EXISTS和IN)的强大的功能。SQL相关的子查询可以在row-by-row的基础上测试外部查询的行与嵌套子查询的行。即使有这些能力，但也还没有复杂到很容易处理所有可能在数据库应用程序设计中遇到的循环情形。循环控制结构提供了检索数据库中的行的能力，也提供了可以用更多的SQL语句处理它们的能力。这比嵌套查询或连接一个单一的SQL语句要更简单、更快速。循环也使得可以在应用程序级完成许多SQL不适用的标准程序设计任务。

注意 PL/SQL也提供了非条件分支语句(带语句标号的GOTO)，应该尽量避免使用它，除非它能使程序更清晰。例如，在冗长的过程中，有时可能会有非常复杂的选择和循环结构，如果简单地使用GOTO来代替，可能会使程序更容易理解。这种情况很少见。例如，简化错误处理的标准情况，通过分支到ERROR标号比异常(exception)处理要简单，参见第12章。

11.2.1 条件控制

PL/SQL为程序执行的选择结构提供了IF-THEN、IF-THEN-ELSE和IF-THEN-ELSIF-ELSE结构。IF-THEN结构检测条件，如果条件成立，则执行一个代码块。例如，下面的触发器块检测当前工人的年龄并显示一个警告窗(alert)，然后通过引发FORM_TRIGGER_FAILURE异常导致触发器失败。

```
DECLARE
  vAlertButton NUMBER;
BEGIN
  IF :Person.Age < 16 THEN
    vAlertButton := Show_Alert('Under_Age_Alert'); --Display error
    RAISE FORM_TRIGGER_FAILURE;
  END IF;
END;
```

注意 关于RAISE语句的更多信息参见第12章“异常和异常处理”部分。

如果有两个程序块，一个在条件判定为TRUE时执行，另一个在条件判定为FALSE时执行，可以使用IF-THEN-ELSE结构：

```
DECLARE
```

China-pub.com
下载


```
    vAlertButton NUMBER;
  BEGIN
    IF :Person.Age < 16 THEN
      vAlertButton := Show_Alert('Under_Age_Alert'); --Display error
      RAISE FORM_TRIGGER_FAILURE;
    ELSE
      vAlertButton := Show_Alert('Over_Age_Alert'); --Display error
    END IF;
  END;
```

如果判定条件有几个部分，可以在 ELSE子句中使用嵌套的IF语句，但更好的方法是使用ELSIF结构：

```
  DECLARE
    vAlertButton NUMBER;
  BEGIN
    IF :Person.Age < 16 THEN
      vAlertButton := Show_Alert('Under_Age_Alert'); --Display error
      RAISE FORM_TRIGGER_FAILURE;
    ELSIF :Person.Age > 70 THEN
      vAlertButton := Show_Alert('Over_Age_Alert'); --Display error
        RAISE FORM_TRIGGER_FAILURE;
    ELSE  -- Worker is between 16 and 70 years old, inclusive
    vAlertButton := Show_Alert('Age_OK_Alert'); --Display info alert
    END IF;
  END;
```

在这个选择结构中，第一个条件识别年龄小于 17的值，第二个条件识别17以上(包括17)并且在70以上的值，其ELSE子句识别17到70之间的值，包括17和70。ELSIF子句全部过滤掉了前面的IF条件和ELSIF子句条件计算为假的情况。

注意 检测NULL值的规则与SQL相同。记住NULL不是一个值，而是不存在数据。因此，必须分别地使用IS NULL和IS NOT NULL表达式代替相等比较，这样来检测某些表达式的值是否为NULL。

11.2.2 重复控制

Oracle Developer嵌入了相当多的循环控制结构。Oracle Developer Forms通过程序块自动地检索记录组，在所有的级别上执行触发器。Oracle Developer Reports用一种非常复杂的循环结构在行和组上循环，报表数据模型通过它的组来驱动这种循环结构。Oracle Developer Graphic也采用循环过程取数据到转换到图表的表中。在触发器代码和程序单元中使用循环之前，应该搞清楚那儿已经存在什么。例如，可以把程序语句放到 item(项)和record(记录)触发器中，Forms将以循环结构自动地执行它们。这比检索所有的记录然后在 PL/SQL块中对它们进行循环要高效，而且更容易维护。可以在报表中把语句放到报表的 page(页)触发器中，Reports将在每一页之后执行它们。PL/SQL控制结构在特定的触发器中提供了几种附加的控制，但是可能的话，应该尽量避免复杂性。

在PL/SQL中有三种循环控制结构：LOOP、WHILE和FOR。

LOOP语句，与EXIT WHEN语句一起使用，执行循环直到指定的条件为真。例如，在一个块中对所有的记录循环，每次age增加1，可以使用如下程序块：

```
  DECLARE
    vCurrentPosition  VARCHAR2(10) := :SYSTEM.CURSOR_RECORD;
```

China-pub.com 下载

```
BEGIN
  FIRST_RECORD;
  LOOP
    :Person.Age := :Person.Age + 1;
    EXIT WHEN :SYSTEM.LAST_RECORD = 'TRUE';
    NEXT_RECORD;
  END LOOP;
  Go_Record(vCurrentPosition);   -- Reset cursor
END;
```

这种结构至少要执行循环语句一次，因为EXIT WHEN的检查在循环语句的后面。如果第一个记录是一个age为NULL的新记录，age的结果将会为NULL，即使经过一次LOOP循环，其值也不会改变。

如果在LOOP语句中嵌套LOOP语句，则应该为LOOP做标号，并在EXIT WHEN中引用标号，以便指定要退出嵌套堆栈中的哪个LOOP。

WHILE语句通过在执行循环语句之前检查条件来控制处理过程：

```
DECLARE
  vCurrentPosition  VARCHAR2 (10) := :SYSTEM.CURSOR_RECORD;
BEGIN
  FIRST_RECORD;
  WHILE :SYSTEM.LAST_RECORD != 'TRUE' LOOP
    :Person.Age := :Person.Age + 1;
    EXIT WHEN :SYSTEM.LAST_RECORD = 'TRUE';
    NEXT_RECORD;
  END LOOP;
  :Person.Age := :Person.Age + 1;
  Go_Record(vCurrentPosition);   -- Reset cursor
END;
```

在条件直接为假时，这种结构根本不执行控制结构中的语句。因此，例子中永远不会更新最后一条记录，所以在循环之后需要对最后记录做更新。和前面一样，如果 age为NULL，语句将会计算并赋值NULL给age,结果没有改变。

最后，FOR语句可以精确地控制循环的次数：

```
DECLARE
  vCurrentPosition  VARCHAR2(10) := :SYSTEM.CURSOR_RECORD;
  vLastRecord VARCHAR2 (10) := 0;
BEGIN
  LAST_RECORD;
  vLastRecord := :SYSTEM.CURSOR_RECORD;
  FIRST_RECORD;
  FOR I IN 1..To_Number(vLastRecord) LOOP
    :Person.Age := :Person.Age + 1;
    NEXT_RECORD;
  END LOOP;
  Go_Record(vCurrentPosition);   -- Reset cursor
END;
```

这种结构在指定的数值范围内执行循环。它也给出了通过在 FOR语句中声明的计数器中使用的值。可以使用该值来查找指定的记录、插入递增值的结果到数据库中，等等。

像例子中一样，可以用表达式指定范围的下限和上限，可以使用这一特性用循环外部的变量来约束循环范围，或者在运行时动态地确定循环范围。例如，通过 SQL语句取得表中的行数，然后按照行数循环读取所对应的行。然而，因为可以对游标使用更有效的% NOTF-

China-pub.com
下载

OUND属性(参见11.3节“在PL/SQL中使用SQL”)，所以这不是使用循环最有效的方式。

例子中通过设置cursor到末记录来取得范围的终值，并保存记录数。这使得可以按照正确循环次数对记录执行循环。与LOOP循环的例子一样，如果只有一条New记录，则将被更新为NULL。

这三个例子中，尽管LOOP循环可能是最直接的，但没有哪一种结构比其他结构更好。通常可以使用三种循环结构来作同样的事情。这样将会判断出那一种的代码行更少，并且判断出那一种结构更简单。作出上面判断的一种方法是看在代码正确的前提下，代码的简单程度：要考虑问题有多少？是单一判断还是复合判断？是否存在要考虑的特殊情况？例如，前面的WHILE代码必须在循环后使用一条语句来更新最后的行，看起来这好像将一件事分成为太多的部分。

注意 在“在PL/SQL中使用SQL”部分介绍FOR循环自动处理游标提取记录的特别版本。如果这种结构的功能能够完成想要做的事，那么它可以节省很大的程序量。

11.3 在PL/SQL中使用SQL

有三个原因要在Oracle Developer应用中使用PL/SQL：在事件处理中使用控制结构改变应用程序的行为；增加特别目的代码实现计算 (除了能够完成的计算项目之外)；在Oracle Developer工具提供的自动功能(很少使用)之外对服务器发出SQL语句。Oracle Developer以服务器上的打包过程的数据块为基础，完全不同于基于表或视图上的数据块。

注意 在PL/SQL中的SQL错误处理采用异常方式。第12章详细讨论了产生和处理异常，以及在错误处理中使用SQLCODE和SQLERRM内部函数。

11.3.1 简单的SQL

在PL/SQL中使用SQL最简单的方法是仅仅使用SQL。PL/SQL把SQL语句看作程序设计语言语句，并且接受几乎是任意的标准 SQL。许多触发器中包含简单的 INSERT、UPDATE或DELETE语句，也许能在应用程序中用宿主变量连接语句和 item。

在PL/SQL中有的SQL语句不能使用：

- 隐含事务结束的DDL语句(对于某些例外，参见11.3.4节“数据定义SQL”)，如CREATE TABLE、GRANT或ALTER VIEW
- 会话控制命令，如SET ROLE
- ALTER SYSTEM
- EXPLAIN PLAN

在触发器中应该避免使用在提交数据到数据库之后能够修改应用的 DML语句(INSERT、UPDATE和DELETE)，因为这样会导致不正确的结果。

警告 可以在PL/SQL程序中使用COMMIT(提交)、ROLLBACK(回滚)和SAVEPOINT(保存点)语句，但是对客户端代码来说这并不是一个好的办法。Oracle Developer运行系统对于事务处理有特殊的方法， COMMIT或ROLLBACK有可能中断事务处理。 Oracle Developer Forms不把COMMIT解释为隐式调用COMMIT_FORM过程，但是应该只在非常高级别代码中，在程序中明确的地方执行。因为 Oracle Developer事务处理很谨慎，

China-pub.com 下载

所以可以避免自己来完成令人头疼的维护工作。

一个特别的SELECT语法可以检索一个单行数据：隐式游标或单行选择。 ANSI(美国国家标准协会)标准要求SELECT INTO语句只返回一行，如果语句找到的符合标准的数据多于一行，则必定产生错误。

隐式游标在选择列表之后加上了一个INTO子句，其后的宿主变量映射选择列表中的元素。可以用%TYPE说明这些变量，或使用那些正确映射数据的项。

```
DECLARE
  vDBAge NUMBER := 0;
BEGIN
  SELECT Age INTO vDBAge FROM Person WHERE Name = :Person.Name;
  -- Use vDBAge to do something here.
END;
```

注意 INTO子句把PL/SQL变量作为用户标识符，而不是宿主变量，所以不用在变量名前加上冒号(:)。宿主变量:Person.Name指Person数据块中的Name项，而不是一个PL/SQL变量。

警告 使用INTO子句和隐式游标应该小心。每个这样的语句在网络中产生两次往返，其中一次用来确定只返回一行，另一次则返回这一行。可以用一个显式游标和提取回数据到变量中的FOR循环来代替隐式游标。这需要更多的代码，会更复杂，更难以维护，所以不得不做一个选择。通常只有在能够确定对性能的影响很小的时候，才使用隐式游标。如果在Internet上开发应用，最好不要使用隐式游标。

11.3.2 使用显式游标

当需要进行查询并且返回的数据多于一行时， PL/SQL中重要的SQL工作开始了。为了完成这项工作，必须声明并使用显式游标。显式游标是一个描述来自 SELECT语句的查询结果的PL/SQL数据结构。PL/SQL提供了许多工具使游标使用更容易。

注意 记住Oracle Developer Form中的块和块的记录以及报表中的数据模型自动处理这种事情。如果要开发描述重复的块或数据模型，可以使用这些自动的工具。显式游标尽管在一般PL/SQL编程中非常有用，但是在Oracle Developer编程中应用较少，因为Oracle Developer自动完成了大多数 SQL任务。可以使用显式游标建立比 Oracle Developer自动建立的更复杂的SQL。代价是必须重写对你无使用的自动功能。

显式游标的形式如下：

```
DECLARE
  CURSOR cSkill IS SELECT Skill, Description FROM Skill;
  vSkill cSkill%ROWTYPE;
BEGIN
  OPEN cSkill;
  LOOP
    FETCH cSkill INTO vSkill;
     EXIT WHEN cSkill%NOTFOUND;
     -- Process the row
  END LOOP;
  CLOSE cSkill;
END;
```

China-pub.com
下载

这两个声明为块建立了基本数据结构。游标包含 SELECT语句从Skill表检索所有数据。记录声明使用%ROWTYPE属性，根据SELECT的结构创建了一个记录。

OPEN(打开)和CLOSE(关闭)语句执行和终止查询。当打开游标时，执行查询，然后必须取回数据行。当关闭游标时，就不能再取回任何数据行了。打开游标后不要忘记关闭。否则将很快达到允许打开的最大游标数。

注意 允许打开的最大游标数有限制，这个限制通过 Oracle数据库服务器 OPEN_CURSORS初始化参数设定。如果从应用程序中作为SQL异常而出现了Forms错误FRM-40515，“ Oracle error:unable to open cursor ”，则需要越过这个参数，或者增大这个参数值，或者在Forms Runtime命令行中设置OPTIMIZETP=NO来共享游标。这会降低处理的速度。共享游标需要重新分析应用程序提交给该服务器的所有 SQL语句。OPEN_CURSORS参数依赖于操作系统，所以在所有的操作系统中有可能无法打开足够多的游标。

LOOP循环显示了提取数据的基本结构。这个游标的% NOTFOUND属性的变异用法测试是否达到了结果表的末尾。这与 SQL消息ROW NOT FOUND一样。

有一个捷径，可以使用 For Loop语句特别的选项实现这个目的：

```
DECLARE
  CURSOR cSkill IS SELECT Skill, Description FROM Skill;
BEGIN
  FOR vSkill IN cSkill LOOP
    null;-- Process the row
  END LOOP;
END;
```

这个块与上面那个较长的块做的事情完全一样。除非在显式游标控制提取、打开游标或关闭游标中需要复杂的逻辑，否则建议使用这种快捷的语法。

注意 记住，如果在SELECT语句中使用表达式，则必须提供一个在记录中能够作为字段名使用的别名。没有别名，无法引用字段。

有几种可以取得游标状态信息的游标属性。表 11-4中的多数属性对显式游标和隐式游标都适用。对显式游标，通过在游标名后加上属性名来使用其属性。对于隐式游标，使用隐式游标名“ SQL ”：例如，SQL%NOTFOUND或SQL%ISOPEN。

表11-4 PL/SQL游标属性

属 性	描 述
%NOTFOUND	当最后一次取数据行失败时为TRUE；使用这个属性检查循环是否已经取得最后一行数据
%FOUND	当最后一次取数据行成功时为TRUE；使用这个属性检查是否取得有效的数据来执行其他进一步的操作，例如插入数据到另一个表
%ROWCOUNT	当打开游标时为0，每取一行数据其值增加1
%ISOPEN	如果游标打开则为TRUE，否则为FALSE；如果有某些控制动作可能已经关闭游标，使用这个属性检测是否游标需要打开

注意 和SELECT语句一样，也可以对数据处理语句和 SELECT语句的隐式游标(INSERT、UPDATE和DELETE)使用这些属性。例如，%ROWCOUNT给出了这些语句的隐式游标的插入、更新或删除的行数。

China-pub.com 下载

可重用游标的一个重要功能是参数化游标的能力。可以在 SELECT语句中任何能够使用宿主变量或字符变量(参见11.1.2节“声明变量和常量”)的地方使用游标参数。当PL/SQL打开游标时，游标接受提供的参数，并把它们插入到 SQL语句。参数的作用域是游标，不能在游标中SELECT语句外的任何地方使用这些参数。下面的代码举例说明了参数化游标的使用：

```
DECLARE
  CURSOR cSkill (pAbbrev IN VARCHAR2 DEFAULT '%') IS
    SELECT Skill, Description FROM Skill
     WHERE UPPER(Skill) LIKE UPPER(pAbbrev);
BEGIN
  FOR vSkill IN cSkill('%Horse%') LOOP
    null; -- Process the row
  END LOOP;
END;
```

这个例子扩展了Skill查询，使其处理基于在 Skill表中模式查找的子集。在例子中使用了LIKE操作符表达式，查询查找出与表达式相匹配的记录。在这个例子中， FOR LOOP在模式中传递，该模式告诉SELECT查找所有包含词语“ horse ”的skill。注意在SELECT中的大写字母转换，这可以进行不区别大小写的比较，使游标的可重用性更好。另外，注意参数中的DEFAULT子句，这个子句提供了在没有参数传递时的显示模式，即“匹配任意字符”模式。

注意 参数化游标在把其作为程序包说明的一部分时十分有用，可以导出游标及其参数。这可以使外部程序使用游标，提供特定目的程序所需的参数。关于程序包的细节参见第12章。

11.3.3 使用游标变量

游标变量是游标的指针。严格地说，游标变量不是变量。在 PL/SQL代码中不能把游标传递给子程序的参数列表，也不能把不同的游标赋给 PL/SQL代码中的单个变量。游标变量可以完成这些。在Oracle Developer中这个特性的主要用途是把游标作为IN OUT变量传递给存储过程。当执行查询的过程在服务器端时，这可以实现在客户端检索数据。

不能仅仅声明一个变量来指向一个游标。首先，必须声明一个类型，然后才能声明变量。声明一个引用游标类型，语法如下：

```
TYPE <type name> IS REF CURSOR [RETURN <record type>];
```

type name(类型名)是用来声明游标变量的名字。在 RETURN子句中可选的记录类型与游标SELECT列表相对应。通常使用% ROWTYPE构造记录类型。如果使用 RETURN子句，PL/SQL检查打开的SQL语句，确保返回的数据与已指定的类型相对应。否则，将使你忙于使SQL与处理语句相匹配。

可以使用变异的OPEN语句打开一个游标变量：

```
OPEN <cursor variable> FOR <select>;
```

OPEN语句把变量与特定的 SELECT(选取)联系起来。作为一个正常的游标，必须在声明部分与其SQL SELECT一起声明游标。作为一个游标变量，应该在执行代码中声明 SQL。

注意 不能在SQL语句中使用变量。这意味着，必须在PL/SQL中为SQL语句编写全部的代码，不能动态地构造和把它传递给游标。可以使用 DBMS_SQL包的动态SQL程序设计能力来动态地分析和执行语句，这个技术超出了本书所涉及的范围。参见11.3.4节

China-pub.com
下载

“数据定义SQL”中使用DBMS_SQL的例子。

下面是一个通过Skill包(关于程序包语法的细节参见第12章)查询与skill相关的信息的例子：

```
CREATE OR REPLACE PACKAGE SkillsPackage AS
    -- A record type with all the columns in the WorkerHasSkill table
  TYPE tSkill IS RECORD (
    vSkill WorkerHasSkill.Skill%TYPE,
    vAbility WorkerHasSkill.Ability%TYPE);
    -- A record type with the primary key column in the Skill table
  TYPE tSkillCursor IS REF CURSOR RETURN tSkill;
    -- A table type for a table of Skills
  PROCEDURE SelectCursor (pcSkill IN OUT tSkillCursor,
                          pPerson IN VARCHAR2);
END SkillsPackage;
```

下面的程序包体用一个简单的SELECT语句打开游标：

```
CREATE OR REPLACE PACKAGE BODY SkillsPackage AS
  PROCEDURE SelectCursor (pcSkill IN OUT tSkillCursor,
                          pPerson IN VARCHAR2) IS
  BEGIN
    OPEN pcSkill FOR
      SELECT Skill, Ability
        FROM WorkerHasSkill
       WHERE Name = pPerson;
  END SelectCursor;
END SkillsPackage;
```

这个包将记录类型、游标类型和创建游标的过程放在了一起。可以通过声明一个记录和游标变量并调用过程来使用它：

```
DECLARE
  vSkill SkillsPackage.tSkill;
  vSkillCursor SkillsPackage.tSkillCursor;
BEGIN
  SkillsPackage.SelectCursor(vSkillCursor, 'Adah Talbot');
  LOOP
    FETCH vSkillCursor INTO vSkill;
    EXIT WHEN vSkillCursor%NOTFOUND;  -- exit after last row fetched
  -- Process the skills of Adah Talbot
  END LOOP;
  CLOSE vSkillCursor;
END;
```

11.3.4 数据定义SQL

在PL/SQL块中能够执行的SQL语句只有SELECT、INSERT、UPDATE和DELETE语句。PL/SQL不允许使用其他嵌入式SQL语句，如：CREATE TABLE、DROP VIEW、COMMIT、GRANT或ALTER SESSION。这些是数据定义(DDL)语句——改变数据库模式或数据字典的语句。

PL/SQL在编译时绑定碰到的SQL对象到所用到的数据字典。系统的这个特性有两个主要的优点：早一点反馈SQL语句的问题和在编译时将SQL对象与其所用的数据字典绑定而带来的性能提高。然而，其限制是编译器不能处理不存在的对象。这意味着PL/SQL不能处理用于创建或破坏数据字典中对象的。

China-pub.com 下载

实际上，这个限制并不很严重。每个 Oracle Developer组件都有一个子程序，可以把任意SQL语句作为其标准程序单元的一部分来执行。另外， PL/SQL提供了支持动态SQL的DBMS_SQL包。

注意 当在应用中使用这些程序单元执行任意SQL语句时，必须清楚它们对事务处理的影响。DDL语句除了影响数据字典之外，会结束当前的事务处理。当执行DDL语句时，Oracle将提交任何未完成的改变给数据库，并移除会话对于数据库特有的任何锁从应用开发角度来看，这意味着必须要小心DDL语句的定时。只有在确保了没有未完成的改变和没有对数据库锁定时，才执行DDL语句。

在Oracle Developer Forms组件中可以使用Forms_DDL过程来执行DDL。例如，如果有一个使用DROP TABLE语句删除一个utility表的菜单项，可以在菜单的 PL/SQL块编写与下面类似的代码：

```
IF :System.Form_Status = 'CHANGED' THEN
  Message('Please save your changes before dropping the utility table.');
ELSE
  Forms_DDL('DROP TABLE UtilityTable');
  IF NOT Form_Success THEN
    MyErrorHandler(DBMS_Error_Code);
  END IF;
END IF;
```

检查表单的状态，确定在执行 DDL语句之前没有未完成的改变。调用 Form_Success内部函数来查看最后语句是否有未完成错误。需要把这段代码传递给用户标准错误处理机构，如例子中的My_Error_Handler过程，这个过程通知用户所发生的事情。

注意 当使用Forms_DDL时，不要用分号终止SQL语句。可以使用Forms_DDL提交一个PL/SQL块，使用标准语法，但不要象在SQL*Plus中那样用斜线终止它。

在Oracle Developer Reports中，使用SRW包中的Do_SQL过程来达到同样的目的。例如，要用下面触发器函数在After Form触发器中删除utility表：

```
FUNCTION DropUtilityTable RETURN BOOLEAN IS
BEGIN
  SRW.Do_SQL('DROP TABLE UtilityTable');
  RETURN (TRUE);
EXCEPTION
  WHEN SRW.Do_SQL_Failure THEN
    SRW.Message(100, 'Error while dropping the utility table.');
    RAISE SRW.Program_Abort;
END;
```

对于Oracle Developer Reports来说，不存在提交未完成改变的问题。主要的区别是错误处理。它不使用函数和返回代码而是使用异常处理来处理问题。当 Do_SQL失败时，引发SRW包异常Do_SQL_Failure。异常处理程序 (参见第12章关于异常处理的细节)输出一个消息，并用SRW内部过程Program_Abort来终止报表。

在Oracle Developer Graphics显示模块中，可以通过内部 Do_SQL过程执行DDL语句。代码与Reports相似，但是没有对于过程的标准异常。

```
PROCEDURE DropUtilityTable IS
BEGIN
  Do_SQL('DROP TABLE UtilityTable');
END;
```

China-pub.com
下载

使用DBMS_SQL程序包代替组件特有的内置子程序来执行 SQL是一个好方法。使用DBMS_SQL提供的一组标准的管理SQL的工具。DBMS_SQL也使用标准SQL错误处理结构，所以可以为数据库的SQL错误编写错误处理代码。

DBMS_SQL的范围太大，本书难以尽述。使用DBMS_SQL的例子如下：

```
CREATE OR REPLACE FUNCTION DropUtilityTable RETURN BOOLEAN IS
  vCursorID INTEGER := DBMS_SQL.Open_Cursor; -- open a cursor for the DDL
  vDummy INTEGER; -- a dummy return value for Execute
BEGIN
  DBMS_SQL.Parse(vCursorID, 'DROP TABLE UtilityTable', DBMS_SQL.NATIVE);
  vDummy := DBMS_SQL.Execute(vCursorID);  -- return value not used for anything
  DBMS_SQL.Close_Cursor(vCursorID);
  RETURN (TRUE);
EXCEPTION
  WHEN OTHERS THEN
    DBMS_SQL.Close_Cursor(vCursorID);
    RAISE;
END;
```

使用DBMS_SQL，首先打开一个显式DDL游标，然后分析与执行语句，最后关闭游标。任何的异常会关闭游标并传递异常(关于异常传播和处理参见第12章)。这个过程非常普遍，可以在客户端或服务器端作为存储函数执行。

第11章介绍了PL/SQL程序设计的基本内容：说明数据结构、编写基本过程代码和把 SQL以相当复杂方式集成到用户代码中。第12章将进入另外一个境界，用PL/SQL组织用户代码。

China-pub.com
下载

第12章　高级PL/SQL

PL/SQL是一种简单的语言，和许多简单的问题一样，简单中也有复杂性。特别情况时，使用第11章中介绍的工具可能创建出一个庞大的拙劣组织起来的代码结构，复制品再三地出现在混乱的应用程序中。

为了减小复杂性，PL/SQL提供了一系列工具来使代码系统变得简单一些。尽管这种结构增加了初始程序任务的复杂性，但是很快将发现，正确使用这些工具会使系统具有更好的可重用性，维护起来也更简单。这些工具包括异常错误的处理、组织大量代码的函数和过程、使用游标和其他数据结构把代码聚集为抽象数据类型的程序包，以及建立重用代码系统的库。

12.1　产生异常

在很多年的旁观之后，“程序异常”逐渐地找到进入主流数据库应用程序设计的道路。任何一个曾经编写过大型数据库应用的人都知道，数据库程序设计中的错误处理代码经常十分晦涩。在现代程序设计语言中，异常是进行错误处理的首选方式。可以用它来代替绝大多数GOTO语句，因为它在程序块中提供了容易理解的控制流程。

异常是程序识出或遭遇异常的条件时，运行时系统引发的一个 PL/SQL信号。这个信号事件有名字，并且有一个处理信号的系统：异常处理程序。

在程序设计中有三种异常可以使用：

- 内部PL/SQL异常：作为PL/SQL程序设计语言一部分的异常。
- 内部的运行时异常：作为Oracle Developer运行时环境(Forms、Graphics和Reports都有不同的异常)一部分的异常。
- 用户定义异常：在代码中定义的处理Oracle错误信息或其他情况的异常。

回忆第11章的PL/SQL块的基本格式：

```
DECLARE
  --Data declarations
BEGIN
  null;--Program statements
EXCEPTION
  --Exception handlers
  WHEN OTHERS THEN
    null;--default handler
END;
```

本部分的重点就在这个程序块中最后的 EXCEPTION(异常)子句上。异常处理的结构与下面所示的子句类似：

```
EXCEPTION
WHEN <exception 1> THEN
  <statements>
...  -- more exception handlers
WHEN OTHERS THEN
```

China-pub.com
下载


```
<statements>
```

EXCEPTION子句由一系列异常处理组成，由 WHEN和异常名(exception name)开头。当程序块产生异常时， PL/SQL停止执行，程序块的EXCEPTION子句检查这个异常的异常处理。如果存在， PL/SQL跳转到该处理，执行后返回到调用者。如果不存在， PL/SQL跳出当前程序块，转到包含当前块的第一个块中的异常处理程序，也就是说，不执行程序块执行的主部分中进一步的程序语句。在产生异常和异常处理程序之间 PL/SQL不执行任何语句。

这种关闭异常的处理方法叫做异常的传播 (propagation)。如果顶层子程序不处理异常，控制传回到调用者的异常处理，并且异常在调用子程序中传播。最后，如果没有处理异常，则异常传播到宿主环境，进行默认处理，通常是回滚处理。 Oracle Developer应用程序将显示一个输出默认信息的警告窗口，然后进行适当的处理。例如，当产生 FORM_TRIGGER_FAILURE异常时， Oracle Developer Forms组件将终止触发器及其驱动的处理过程，通常把游标停留在引起异常的项或记录上。

可以直接通过RAISE语句产生一个任意的异常：

```
RAISE <exception-name>;
```

PL/SQL执行上面的语句时，则把控制转换到当前块的异常处理程序。如果块中没有处理这个异常，则传播异常。

如果要处理异常，但又不希望外部块或调用子程序发现异常，可以直接通过一条 RAISE语句传播异常：

```
RAISE;
```

这应该是异常处理的最后一条语句。

注意 如果在声明部分或异常部分产生异常，则异常立刻传播。所以必须把这个异常的任何异常处理程序放到一个外部块中。

下面的程序包定义了一系列嵌套的子程序，这些子程序将作为异常传播的例子。还在后面给出了程序包结构细节和使用。首先，必须创建程序包的说明：

```
CREATE OR REPLACE PACKAGE exceptionPackage IS
  eTest EXCEPTION;
  PROCEDURE OuterProc;
END exceptionPackage;
```

程序包用一系列嵌套的块和子程序实现 Outer_Proc过程。用块结构的复杂性举例说明异常是如何传播的。 Inner Block#2块出现一个test异常，并且异常通过异常处理程序传播。

```
CREATE OR REPLACE PACKAGE BODY exceptionPackage IS
  PROCEDURE OuterProc IS
  BEGIN -- OuterProc block
    DECLARE -- Start of Inner Block #1
      PROCEDURE InnerProc IS
      BEGIN -- Inner Procedure
        BEGIN -- Inner Block #2
          RAISE eTest; -- explicitly raise the exception here
        EXCEPTION  -- Inner Block #2 exception handlers
          WHEN eTest THEN
            Message('Exception handled in Inner Block #2');
            RAISE;
        END; -- End of Inner Block #2
```

China-pub.com 下载

```
      EXCEPTION  --Inner Proc exception handlers
        WHEN eTest THEN
          Message('Exception handled in InnerProc');
          RAISE;
      END InnerProc;
    BEGIN -- Inner block #1
      InnerProc;
    EXCEPTION  -- Inner Block #1 exception handlers
      WHEN eTest THEN
        Message('Exception handled in Inner Block #1');
        RAISE;
    END;  -- End of Inner Block #1
  EXCEPTION  -- OuterProc exception handlers
    WHEN eTest THEN
      Message('Exception handled in OuterProc');
      RAISE;
  END OuterProc;
END exceptionPackage;
```

下面的代码调用Outer_Proc过程：

```
BEGIN
  exceptionPackage.OuterProc;
EXCEPTION
  WHEN exceptionPackage.eTest THEN
    Message('Exception handled in calling procedure');
END;
```

在Oracle Developer中运行这段代码，产生如下输出信息：

```
Exception handled in Inner Block #2
Exception handled in InnerProc
Exception handled in Inner Block #1
Exception handled in OuterProc
Exception handled in calling procedure
```

Inner Block#2出现异常。其异常处理程序把异常传播到包含 Inner Block#2的InnerProc块中。InnerProc块的异常处理程序又将异常传播到包含它的 Inner Block#1块中,在Inner Block#1中，传播异常到所包含的 OuterProc。这个过程是分层结构的最顶层子程序，所以它把异常传播给调用子程序的块。

注意 当异常传播到子程序外部时，PL/SQL不设置子程序的OUT参数，并且自动地回滚子程序中完成的工作。如果不希望这样，则应该在子程序的外部块编写 WHEN OTHERS子句代码，来按照希望的方式处理异常。

12.1.1 PL/SQL的内部异常

PL/SQL定义了几种异常作为 STANDARD(标准)包的一部分。所有的 PL/SQL程序都能够引发这些异常。需要在代码中处理的公共异常是 NO_DATA_FOUND和VALUE_ERROR异常。当单行的选择 (SELECT…INTO)失败，返回一行时发生 NO_DATA_FOUND异常。VALUE_ERROR异常在语句出现值的问题 (算术、转换、截断或约束)时发生。产生VALUE_ERROR异常的大多数情况是把过长的字符串赋给一个变量。例如，定义一个变量为 VARCHAR2(10)，并把字符串“ George E.Talbot ”赋给它，将引发 VALUE_ERROR异常。字符串有 14个字符，

China-pub.com
下载

而变量只允许有10个字符。

12.1.2 Oracle Developer的内部异常

Oracle Developer的三个组件都有其特定环境的异常。

Form Builder在正常运行处理中提供了 FORM_TRIGGER_FAILURE异常。表单组件在处理的任何一个触发器失败时，产生这个异常。可以在触发器的代码中使用这个异常来告诉PL/SQL触发器失败：

```
RAISE FORM_TRIGGER_FAILURE;
```

Form Builder还提供了DEBUG.BREAK异常，可以在块中通过RAISE语句显式地嵌入调试中断。参见第15章。

Oracle Developer Reports组件把它的异常打包到SRW包中。

在Reports组件中两个最常用的异常是RUN_REPORT_FAILURE异常和PROGRAM_ABORT异常。RUN_REPORT_FAILURE异常表示报表失败，显示一个常规信息并结束报表。可以编写异常处理来显示用户自己的错误信息。在代码中出现致命的错误时，可以产生 PROGRAM_ABORT异常并显示适当的错误信息。

Oracle Developer Graphics组件有一大批内部异常，在此无法尽述。完整的列表参见联机的Graphics Builder参考。如果在Graphics Builder中大量地编写代码，将会发现要使用许多异常处理程序来处理意外情况的发生。在一个 Graphic的显示区有许多能够编程的对象，每个对象都可能有不同的错误发生。大约有 200种这样的异常。在使用 OG和TOOLINT包编程时可能会遇到它们。

例如，试图用OG_UPDATE_CHART显示日期坐标轴,但是日期超过了坐标轴的范围，Oracle Developer Graphics组件产生OG_DATE_OVERFLOW异常。当在坐标轴上指定一个无效的位置时，产生OG_INVALID_POSITION异常。

12.1.3 用户定义的异常

有两种用户定义的异常：Oracle错误异常和用户异常。

可以使用编译指令pragma EXCEPTION_INIT来为某个Oracle错误信息建立指定的异常。pragma是一条告诉编译器在编译时做什么的指令。既然这样，告诉编译器为 Oracle错误代码创建一个异常：

```
ePrivileges EXCEPTION;
PRAGMA EXCEPTION_INIT(ePrivileges, -1031);
```

这条语句把Oracle错误代码-1031与名字为privilegesException异常连接。当核心Oracle数据库服务器引擎发出这个错误代码给 PL/SQL时，PL/SQL产生这个命名异常并把控制传递到用户的异常处理程序。这个 PRAGMA保存某些代码。否则，需要捕捉 Oracle数据库服务器错误条件并为错误数(不需要编写的附加代码)测试错误代码(SQLCODE)。

要定义一个简单的用户异常，只需要在 Declare块中说明异常。例如，如果触发器为总帐记录做内部检验，则可能需要在检验失败时产生一个专门的异常，以便能够报告出这个问题：

```
DECLARE
  eValidation   EXCEPTION;
```

China-pub.com 下载

```
  vAlertButton NUMBER;
BEGIN
   null;  -- Do the validating.
EXCEPTION
  WHEN eValidation THEN
    vAlertButton := Show_Alert('LedgerValidationAlert');
    RAISE FORM_TRIGGER_FAILURE;
  WHEN OTHERS THEN
    RAISE FORM_TRIGGER_FAILURE;
END;
```

异常的作用域与变量相似。可以在不同的作用域中多次定义同一个异常。最好的方法是，仅仅在程序包中一次性定义一个给定的异常，将它导出到使用它的子程序中。可以用不同的名字定义异常，这可以确保处理异常的处理程序能够正确处理它。

注意 如果用同一个名字定义两个异常，则有两个独立的异常。这会使编写错误处理代码时非常混乱，尤其是在子程序有嵌套块时。用一个内置的异常名，如 NO_DATA_FOUND，来说明用户的异常尤其是不明智的。

12.1.4 使用块控制异常处理

在某些条件下，需要在处理异常之后继续原来的进程。但不幸的是，当 PL/SQL产生异常时，PL/SQL结束块的进程并在结束异常处理代码后返回调用者。这意味着不能返回块中引起异常的位置并且继续下去。通常的情况是，要在若干语句中取一个，或者希望即使在执行时由于某种原因引起失败，也要执行全部所有的代码。

如果想在处理过程中有完全的控制权，则必须把子程序分开到嵌套块中。关于块嵌套的详细信息，参见下面关于子程序(过程和函数)部分。需要做的是把每个“不致命”的异常的异常处理程序放到单独的块中。某个异常处理程序执行之后，对于嵌套的块仍然存在并继续执行，只是跳过了嵌套块中的代码。

例如，要浏览一个表中的一个数据，但也希望在原始值不存在时建立一个默认值。需要执行几条SQL语句来确定真实状态，并根据情况分支，结构如下：

```
DECLARE
  vLookupValue VARCHAR2(100);
BEGIN
  BEGIN
    SELECT lookupValue INTO vLookupValue FROM LookupTable;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      vLookupValue := 'N/A';
  END;
  INSERT INTO AuditTable (currentDate, currentValue)
    VALUES (sysdate, vLookupValue);
END;
```

这个块首先执行单行选择来发现查找值。如果服务器没有返回数据，嵌套块异常处理程序赋了一个默认值，并且继续处理在嵌套块中 END下面的第一条语句：INSERT语句。

12.1.5 错误处理函数

有两个PL/SQL内部函数可以在异常处理程序中处理数据库的错误。

China-pub.com
下载

SQLCODE函数返回最后错误的错误代码。这个代码对于在 PRAGMA EXCEPTION_INIT 中定义的错误来说，是一个负数，或者是 +100(NOT FOUND)，或者是+1。

SQLERRM函数是错误信息文本，包含错误代码。

Oracle Developer Forms也提供了一些错误处理内部函数，见表 12-1。

表12-1 Oracle Developer Forms组件的内置错误处理函数

函 数	返回值类型	描 述
ERROR_TYPE	CHAR	返回最后发生的错误的类型：Forms错误为FRM，数据库错误为ORA
ERROR_CODE	NUMBER	返回最后Forms错误的代码
ERROR_TEXT	CHAR	返回最后Forms错误的文本
DBMS_ERROR_CODE	NUMBER	返回Forms检测到的最后数据库错误的代码
DBMS_ERROR_TEXT	CHAR	返回一个或多个数据库错误消息的类型和文本，在其前面是DBMS_ERROR_CODE

12.2 减少使用匿名块

第11章中采用的无名块对触发器建立简单大块代码有很大影响。当需要使用可重用代码时，无名块变得用处不大。应该到更正式的命名块和参数化块的组织机构去进一步讨论。

12.2.1 子程序

PL/SQL提供了两种实现语言组织功能的对象：过程和函数。这两种代码组织都是子程序：是一些有名字的、参数化的块代码，可以用参数来调用它们执行某些操作。函数是计算某个值的子程序，而过程只是执行一个动作。在 Oracle Developer中，可以用它们在程序单元标题下面的各种组件中编写代码，或者把它们添加到库组件中。必须在触发器代码中或其他任何引用子程序名的地方显式地调用子程序 (例如，在报表组件中的组过滤器和公式中使用函数)。调用子程序是一个PL/SQL语句。因此，在建立一个子程序时，是通过增加一种新语句扩展了PL/SQL语言。

通常，在需要计算一个单一值时可以编写函数。在想执行一个操作而不是计算一个值时，应该编写一个过程。然而，所有的子程序都可以有输出参数。因为希望函数返回一个单一值，所以在编写函数代码时不应该使用 OUT或IN OUT参数。对于错误条件和返回及其他类似操作，应该加以定义并截获异常 (见前面的部分)。有关输出参数的更多信息见下面部分。

子程序可以重载。可以在程序包中或另外子程序中或有多个子程序的块中用同样的名字声明多个子程序。细节见本章 12.3节“ 创造性的包 ”部分。

1. 过程

过程如下所示，由名字、一组参数和一段代码组成：

```
PROCEDURE <name> [<argument list>] IS
  <declarations>
BEGIN
  <program statements>
END;
```

除了用 PROCEDURE…IS序列代替了 DECLARE 关键字之外，其语法与无名块相同。PROCEDURE…IS序列是过程说明，其余的部分是过程体。可以在调用子程序的子程序或块

China-pub.com 下载

之前只编写说明代码，然后在后面的代码中编写完整的子程序。

2. 函数

函数由名字、一组参数、返回值类型和一段代码组成。当从函数返回时，函数返回给调用者一个指定类型的值(这种语法与过程的相似，除了说明中的RETURN子句)。

```
FUNCTION <name> [<argument list>] RETURN <type> IS
  <declarations>
BEGIN
  <program statements>
END;
```

为了完成函数的处理并返回函数值，使用的 RETURN语句应该包含为返回值赋值的表达式。

注意 如果你高兴可以有很多个RETURN语句，但是这是一个坏习惯，RETURN语句最好不要超过一个。有两个或更多的RETURN语句将导致程序控制的可能路径增多，使程序单元的测试变得困难。当进行集成测试时，必须去检验每个调用者的返回值以确定调用者取得了正确值。因此，多个返回节省了一些代码，却导致测试工作量和潜在缺陷的巨大增加。可以在子程序中使用内部异常，并从异常处理程序返回。一般来说，尽管如此，在编写代码时最好避免这种情况。可以通过嵌套块来实现这个目的 (参见12.2.2节“嵌套块”)。

3. 参数

通过在括号中的一个说明列表来为子程序定义一组参数：

```
( <variable> [ <mode> ] <type> [ := | DEFAULT <value>, ... )
```

<variable>是标准PL/SQL标识符，表示参数名。<mode>是如下三种可能性的一种：

- IN：这个参数的值是子程序的输入值，给该变量赋值不影响其在调用程序中的值。如果不指定方式，这是默认的方式。这种参数可以是变量、常量、直接量或表达式。
- OUT：这个参数用于从子程序中输出，该变量用来赋一个子程序返回时调用者能够使用的值。在没有赋值之前，不能使用这个变量。该变量在调用者中必须是能够修改的变量，并且必须是变量而不是直接量，因为不能给一个值赋值。
- IN OUT：这个参数是既能用于输入又能用于输出的变量。可以使用其输入值，又能给它赋一个调用者能够看到的输出值。因为要给参数赋值，这种参数必须是变量。

另外，在函数中应该避免使用OUT或IN OUT参数，因为函数通常只返回一个唯一的返回值。

<type>可以是任意的有效PL/SQL类型，包括Oracle Developer对象或自定义的类型。不需要指定类型的刻度或精度，例如VARCHAR2(20)，只使用类型名。

“:=IDEFAULT<value>”子句表示既可以使用赋值操作:=，也可以使用DEFAULT关键字给参数赋一个默认值。这可以使得可以不带参数而调用子程序，在这种情况下，子程序给参数赋默认值。这种功能只能用于IN变量，不能用于OUT或IN OUT变量。

列表中参数之间用逗号分隔。

如果子程序没有参数，则该子程序没有括号：

```
PROCEDURE <name>;
FUNCTION <name> RETURN <type>;
```

China-pub.com
下载

4. 调用子程序

子程序调用是一条PL/SQL语句，可以在放置PL/SQL语句的任何地方放置这样一条调用语句。调用子程序只需要块中的一条语句：

```
Do_Key('NEXT RECORD');
```

可以把子程序的调用作为无名块的一部分或过程的一部分：

```
DECLARE
  vAlertButton NUMBER;
BEGIN
  vAlertButton := Show_Alert('WarningAlert');
END;
```

提示 当调用一个函数时，如前面例子所示，必须把返回结果赋值给一个变量。如果不这样，将出现PL/SQL找不到过程的提示。PL/SQL如何确认一个函数与返回类型无关。

在哪里说明的子程序决定了在哪里能够调用它。如果在一个模块 (表单、菜单、报表或图形显示)中定义了一个子程序，只有这个模块的触发器和其他子程序能够调用这个子程序。例如，如果在表单模块中定义了一个过程，只有表单的触发器和表单子程序能够调用这个子程序。如果在菜单模块中定义了一个函数，只有在菜单项命令和初始化代码中能够调用这个函数。

如果在库模块中定义了一个子程序，必须首先把库连接到模块，才能调用这个子程序。为了实现这个目的，在模块中找到 Attached Libraries标题，单击Create按钮，通过结果对话框选择连接库。

调用存储的子程序(在服务器上定义的子程序，而不是在库或模块中)稍微复杂一点。一个好方法是，首先为子程序定义一个公共同义词。有了这个同义词，调用者只须指定子程序的名字即可。

另外，必须有适当的权限来执行过程。如果用过程的所有者身份登录，则自动取得EXECUTE(执行)权限，所有者必须给你的用户或角色授予这种权限：

```
GRANT EXECUTE ON Talbot.AddLedgerTransaction to AccountingRole;
```

现在，任何被授权AccountingRole角色的用户都可以执行AddLedgerTransaction子程序。

注意 如果在调用子程序，且PL/SQL不能识别子程序名，应该仔细检查过程说明的类型和参数个数。PL/SQL使用子程序名、返回类型(对函数而言)和参数的存在及类型来识别一个特定的子程序。如果有一项不存在，或者指定的类型不正确，或者传递的参数类型不正确，则PL/SQL不能确认对被调用子程序的调用。如前面所述，如果使用一个函数，要确定把返回值赋值给了一个变量，因为PL/SQL使用它来确认函数名。

按位传递 传递参数给子程序有两种途径。第一种途径是按位传递——按子程序定义参数的顺序传递参数值。例如，说明过程如下：

```
PROCEDURE testLedgerItem(pItemID NUMBER, pAmount NUMBER);
```

过程调用如下：

```
testLedgerItem(456, :Ledger.Amount);
```

值“456”传递给pItemID，表单项Ledger.Amount传递给pAmount。注意，传递项目对象值(item)使用绑定变量的语法(冒号前缀)。

按名传递 另外一种途径是按名传递。使用这种语法，可以按任意参数名的顺序传递参

China-pub.com 下载

数值：

```
testLedgerItem(pAmount => :Ledger.Amount, pItemID => 456);
```

按名传递对默认值尤其有用。如果在有默认值时使用按位传递，可以省去有默认值的尾随参数。如果使用按名传递，可以省去任何有默认值的参数，不仅仅是尾随参数。这意味着，可以省去某些有默认值的参数，但要包含其他在参数定义中没有默认值的参数。

5. 报表过滤器和公式

在Oracle Developer Reports中函数的一个主要用途是实现过滤器和公式，细节参见第 7章。

报表组过滤器是一个返回Boolean值的函数，Oracle Developer用它来决定在组中显示哪些记录。通过设定组过滤器属性的 Condition(条件)和提供一个函数，告诉 Reports只显示组中那些函数值为TRUE的记录。

报表公式是一个函数，它利用类型属性公式为分组中的每个记录返回一个值，并附加到一个数据列中。可以使用一个表达式来代替这个函数，但是那样就不能使用函数所使用的PL/SQL的完整功能。函数返回一个与列数据类型项对应的单一值：数值列的函数返回值为NUMBER类型；日期列的函数返回值为 DATE类型；字符串类函数的返回值为 CHARACTER、VARCHAR或VARCHAR2类型。

12.2.2 嵌套块

PL/SQL的块结构本性提供了在作用域方面对子程序部分的强大控制。程序的块结构是使用PL/SQL块(DECLARE…BEGIN…END;序列)来组织变量有效性和异常处理程序的途径。

块结构的基本概念是一个块是一条程序语句。这条语句按顺序执行，像任何程序语句一样。然而，在语句中可能有与外部块及其进程不知道的事情有关的庞大的复杂结构。隐藏 (或者封装)可以使内部块的复杂工作与外部块的顺序无关。这样减小了程序的复杂性，使程序更容易理解、调试、测试和维护。例如，可以推迟嵌套块体的代码编写，使程序残留一些不完全的地方，然后可以继续编写代码和测试子程序的其他部分。在需要用到嵌套块之前，不需要考虑嵌套块的细节。

在前面关于异常处理的部分中提供了一个使用嵌套块控制异常处理结束之后执行流程的例子。通过把异常处理程序放到一个内部块中，可以实现在异常处理之后继续外部块的处理。

块结构也能够在子程序中隔离代码的逻辑簇。例如，需要一个有多个返回的子程序，则应该考虑嵌套那段不想执行的代码，并且有条件地执行那个块。然后子程序就有唯一的退出点。这可以使程序的控制结构清晰，尤其对于测试有用。

通过说明嵌套块，可以在块间共享数据元素而不需要使用全局变量。嵌套块可以访问外部块说明的所有变量。

可以通过在外部子程序说明部分包含子程序，把子程序整个嵌套到其他子程序中。必须在所有的其他说明(变量、类型或异常)之后声明嵌套子程序。

12.2.3 在导航器中定位块

现在知道了如何编写子程序。下面的问题是在什么地方、如何编写子程序。前面讲述了两个可以编写子程序的地方：

■ Program units(程序单元)：在表单、菜单、显示或报表模块的 Program Units标题之后。

China-pub.com
下载

■ Library(库)：在Forms、Reports或Graphics的库中Libraries 标题之后。

也有第三种选择：

■ Stored(存储)：通过CREATE FUNCTION和CREATE PROCEDURE语句或者通过在Database Objects标题下建立子程序存储在数据库中。

如何确定在哪儿定义子程序？

如果使用子程序访问那些不希望普遍可用的数据，则使用存储子程序。这是个封装过程，在6.2.5节“过程的封装”中讨论。如果在子程序中完成许多数据库工作，也使用存储子程序。这可以使调用子程序和服务器执行每件事更有力 (大概)。这也可以减少网络传输。如果操作是对数据库对象的常规操作，使用存储子程序。这可以与对象的所有用户共享操作。这是结合性原则，把运算存储到其操作的数据附近。

注意 因为服务器和客户机版本的 PL/SQL可能有差别，可以建立用服务器端版本PL/SQL封装函数的存储过程供客户端版本使用。例如，如果对Oracle 7和Oracle 8使用Oracle Developer，就不使用同样版本的PL/SQL处理。可以使用DBMS_SQL建立存储子程序，适当地使其参数化，然后从客户端PL/SQL调用它们。当运行Oracle 7数据库时，在服务器上运行Oracle 7 PL/SQL；当运行Oracle 8数据库时，运行PL/SQL8。

如果想要使子程序被多于一个的模块使用，则使用库。这通常适用于适应不同情况的常规操作。另外，子程序应该只在几个有限的方式下适用于数据库。存储子程序用于数据库相关操作很多的情况下，可以减小网络传输和使得重使用更加容易。有一种情况例外，这就是子程序需要调用组件的内部子程序。因为无法从数据库服务器调用 Oracle Developer内部子程序。另外一种例外情况是子程序需要引用 Oracle Developer模块的变量。然而，可以把这些值作为参数传递给子程序，而不是直接引用它们。这种子程序和数据模块的松耦合是一个优良的系统结构的基本原则。这种送耦合意味着可以把子程序存储到数据库并把组件模块传递给它。

本章最后的“检查库”节讲述了如何在库中建立子程序。一旦编写了子程序代码，就可以把库附加到任何模块并在模块中使用子程序。要附加一个库，在模块中找到 Attached Libraries(附加的库)的标题，选择它，单击Create工具。然后就可以通过一个标准的 Open File对话框选择附加库。不要忘记了从文件名中移除所有的路径信息。

子程序在模块中。当代码对模块十分特殊，在其他模块中没有任何用处时，应该把子程序放在模块中。模块大量使用项、大量使用应用程序专用的单独逻辑关系、通过内部子程序对其他组件的大量使用，这些都表明模块的特殊性。

要建立一个模块程序单元，在模块下找到 Program Units(程序单元)的标题，单击Create工具，会看到到一个 New Program Unit对话框，使用同样的 PL/SQL Editor定义子程序。但是Oracle Developer分别用表单、菜单、显示或报表模块存储代码。后面的“建立库”部分详细讲述了如何使用这些工具建立子程序和包。

另外一种情况是定义包含子程序操作的对象组。这种对象组的行为好像是一种程序包，它使得在程序包中的子程序在其他模块中的重用有意义。这是 PL/SQL包或库的选择对象，在后面的部分将介绍更多的内容。

当Oracle Developer寻找程序单元时，首先检查模块的 Program Units部分。其次，在模块中Attached Libraries部分连接的库的程序单元中查找。第三，在数据库服务器上查找。使用这

China-pub.com 下载

个顺序的优点是有助于在分级结构的高层使用同样的名字添加程序单元。例如，有一个库子程序支持大多数应用程序的默认行为。在模块的 Program Units部分添加了一个与其名字一样的子程序。当Oracle Developer运行应用程序并调用子程序时，则首先查找添加的子程序，决不会执行库子程序。

12.3 创造性的包装

程序包可以将其他PL/SQL元素联合到一个独立的整体中：一系列数据对象、类型、游标、异常和子程序。程序包有说明和程序体。程序包的说明是它的公用面：说明它所使用的元素。这些元素是公用元素。包的程序体是游标和所定义的子程序的实施，也包含要说明的私有元素。可以替换包程序体而不改写包的说明。

注意 不能在程序单元中嵌套程序包。

12.3.1 创建程序包

为什么要创建程序包？程序包提供了一种方式来将 PL/SQL元素结合到一个系统中。程序包与对象定向系统的对象相比：程序包中包含与系统中某些结合的部分相关的状态和行为。程序包提供了完整的数据抽象和信息隐蔽，从而提高了系统的模块性。这种改善了的内聚性和数据抽象之间的松耦合，又直接提高了代码的可读性和软件的维护能力。另外，一旦要执行程序包中的模块，Oracle将把程序包加载到内存中。所以使用程序包组件的进一步处理会执行得更加好。

例如，可以改变程序包中的函数而不用改变任何调用这个函数的调用者，因为接口说明没有改变。也可以使一个子程序成为存根程序 (stub)直到为它们编写代码为止，甚至可以根本不为其提供程序体。这可以在程序包还没有真正完成的情况下对使用程序包的程序单元进行编译，进行可能的单元测试。

程序包也能够定义变量、常量和在整个会话中持续的游标为公有元素或私有元素。提供了在过程调用或事务处理之间保持信息而不依赖于无特定结构的全局变量的能力。通过程序包，可以使数据和操作的控制结构化。所有这些元素都是 NULL ，除非在说明中初始化它们。可以定义子程序完全在程序包的程序体中。那样，就不能在程序包以外访问这些子程序。

最后，当使用程序包的任何一个部分时，PL/SQL的程序包技术加载整个程序包到内存中。这意味着对程序包中程序另外的调用不需要另外加载代码，提高了性能。这个功能也说明，程序包要相对简单并且要内聚，而不是把一大堆不相干的代码集合到一个程序包中。

有三种内聚表明应该使用程序包：

- 功能内聚：一组元素作为一个整体，描述应用的一个特定的功能，例如一个对象类型或与簇相关的对象类型。
- 抽象内聚：这是一组程序单元，它们作为提供一组定义明确的服务的抽象层在一起工作。一个应用程序界面或 API(应用程序接口)表现出这种结合，其例子包括 Oracle Report Builder的SRW包和Graphics Builder的OG包。
- 实用性内聚：在一起使用的一组对象与 /或程序单元，在其中没有服务或功能的使用，是实用性结合。一个特殊的实用性内聚的情况是事务处理及其元素 (无论如何不同)打包到一个游标和子程序包中。

China-pub.com
下载

定位程序包的逻辑与子程序相同 (如果必要，回顾本章前面“在导航器中定位块”部分)。如果程序包结构建立得好，很容易区别数据库相关程序包和纯粹的客户端定向程序包。由于重用性的缘故，大多数程序包应该在库中开发，而不是在模块中开发。有一个例外是像块或图表那样的模块对象聚合一个程序包。建立程序包的过程与子程序的建立完全相同。参见本章“子程序”部分。

程序包有两种语法，一种是程序包说明语法，另外一种是程序包程序体语法。

1. 程序包说明

```
PACKAGE <name> IS
  {<variable spec>|<type spec>|<cursor spec>|<exception spec>} ...
  <subprogram spec> ...
END;
```

关于变量和类型说明的细节参见第 11章。说明包括一组对象说明 (变量、常量、类型、游标和异常)和一组子程序说明。

游标按如下格式说明：

```
CURSOR <name> <parameter list> RETURN <type>;
```

游标说明，因为不包括SELECT语句，必须提供RETURN子句来识别游标取回记录的类型。<type>可以是下面的任何一种类型：

- Record：先前定义的记录类型。
- Variable%TYPE：先前定义的变量类型。
- Table.Column%TYPE：数据库列类型。
- Table%ROWTYPE：一个表中所有列的记录类型。

如果要做的不仅仅是从表中选择列，例如 SELECT序列中包含GROUP BY或者表达式，则必须提供记录类型。这个类型必须与程序包程序体中游标体中定义的类型相匹配。游标说明的目的是定义游标的接口，游标接口由游标名、参数列表和返回类型组成。

子程序说明语法如下：

```
PROCEDURE | FUNCTION <name> <parameter list>
  [RETURN <type>];
```

参数列表的语法参见前面的“参数”部分。 RETURN子句只在函数中提供。在一个完整的子程序定义中，语句在IS和子程序体之后继续。子程序说明的目的是导出子程序的接口，程序的接口由子程序名、参数列表和返回类型组成。在程序包说明中，子程序的说明在其他说明之后。在程序包中子程序可以重载，可以在程序包中或另外的子程序中或块中用同样的名字说明多个子程序，只要参数的数目与 /或参数类型不同。这可以建立通过不同参数可选择的子程序。一个例子是Oracle Developer内部程序包中的内部子程序，其第一个参数既可以是对象名，也可以是对象标识符 ID。在内部程序包(Forms STANDARD、Reports SRW或Graphics OG)中确实有两个独立的、重载的子程序。子程序的重载可以为不同用户建立一组非常灵活的操作，使程序包的可重用性更大，使用更容易。

注意 类型必须完全不同。REAL和NUMBER及FLOAT类型在考虑子程序重载时，它们是相同的。这适用于函数的RETURN类型。

下面程序包说明的例子是处理 24小时时间值的程序包的接口说明。程序包包含一个记录类型、一组异常和对记录类型操作的算术函数。这个程序包引用另外一个程序包 TimeInterval-

China-pub.com 下载

Package，这个程序包导出一个时间间隔类型 (TimeIntervalType)和处理这种类型的函数。为了方便指定的时间和时间间隔，按任意先后顺序相加，重载了算术加法运算符。减法是不对称的。

```
CREATE OR REPLACE PACKAGE Time24Package IS
  TYPE tTime24 IS RECORD (
    vHour NATURAL NOT NULL := 0,
    vMinute NATURAL NOT NULL := 0,
    vSecond NUMBER NOT NULL := 0.0);
  eInvalidHour EXCEPTION;
  eInvalidMinute EXCEPTION;
  eInvalidSecond EXCEPTION;
  FUNCTION CurrentTime RETURN tTime24;
  PROCEDURE SetTime(pTime IN OUT tTime24, pHour IN NUMBER,
    pMinute IN NUMBER, pSecond IN NUMBER);
  FUNCTION Add(pTime IN tTime24,
               pInterval IN TimeIntervalPackage.tTimeInterval)
    RETURN tTime24;
  FUNCTION Add(pInterval IN TimeIntervalPackage.tTimeInterval,
               pTime IN tTime24)
    RETURN tTime24;
  FUNCTION Subtract(pTime IN tTime24,
                    pInterval IN TimeIntervalPackage.tTimeInterval)
    RETURN tTime24;
  FUNCTION Format(pTime IN tTime24) RETURN VARCHAR2;
  PROCEDURE Validate(pTime IN tTime24);
END Time24Package;
```

这是描述一个对象的程序包的例子。 TYPE说明不建立真实记录，只定义记录的类型。函数都引用了一种或两种记录类型。程序包自己本身不包含数据。在使用程序包的子程序中建立数据对象、记录类型变量。然后调用与该变量有关的函数完成对变量的操作，或者通过返回一个新的对象类型建立新数据。记录的个体元素作为说明的一部分对于调用者是可见的。忽略它而对数据操作只使用函数是一个好的习惯。

注意 在内存中程序包只存在一份。这表示如果把变量作为程序包的一部分定义，那么只有一个这样的变量。如果这能够满足需要，那最好了。上面的例子很清楚地表明在那里它是不合适的。有可能不得不建立多种时间值，而不仅仅是一个。这意味着不能完全把记录类型结构封装到程序包中——至少是没有大量的程序编制、间接处理、艰难的维护。这是一个在哪里使用比较好而不是坚持“按正确的方式去做”的事例。可以尽可能容易和少付出努力去做想要做的事，除了冒滥用权限直接访问结构的危险之外。

2. 程序包程序体

程序包程序体语法如下：

```
PACKAGE BODY <name> IS
{<variable spec>|<type spec>|<cursor body>|<exception spec>} ...
  <subprogram body> ...
[BEGIN
  <statement>; ...
[EXCEPTION
  <exception handlers> ...]]
END;
```

China-pub.com
下载

只有游标和子程序体与在程序包说明中说明的子程序相对应。其余的说明 (变量等等)对于程序包的程序体都是私有的，不能在外部代码中使用这些元素。

游标体的语法与游标说明的语法相似，除了附加的 SELECT语句：

```
CURSOR <name> <parameter list> RETURN <type> IS <SELECT statement>;
```

参数列表必须与游标说明中的参数列表精确匹配， RETURN类型也是这样。

程序包程序体初始化在程序包程序体的末尾随意的 BEGIN…END序列之间，在说明部分定义完所有游标和子程序元素之后。这块代码只在第一次引用程序包元素时执行一次，使用它来初始化程序包的私有或共有元素。异常处理程序处理初始化代码引起的任何异常。

下面的例子显示了前面部分定义的程序包的程序体：

```
CREATE OR REPLACE PACKAGE BODY Time24Package IS
  cSeparator CONSTANT CHAR(1) := ':';  -- format cSeparator hh:mm:ss
  cSecondModulus CONSTANT NUMBER := 60.0;
  cMinuteModulus CONSTANT NATURAL := 60;
  cHourModulus CONSTANT NATURAL := 24;
  cSecondsPerMinute CONSTANT NUMBER := 60.0;
  cSecondsPerHour CONSTANT NUMBER := 60*cSecondsPerMinute;
  cSecondsPerDay CONSTANT NATURAL := 24*cSecondsPerHour;
  eSecondsOverflow EXCEPTION;  -- internal error

  -- An internal Convert function to convert time values
  -- to and from seconds
  FUNCTION Convert(pTime tTime24) RETURN NUMBER IS
    vSeconds NUMBER := 0;
  BEGIN
    Validate(pTime);
    vSeconds := (pTime.vHour * cSecondsPerHour) +
                (pTime.vMinute * cSecondsPerMinute) +
                 pTime.vSecond;
    IF vSeconds NOT BETWEEN 0 AND cSecondsPerDay THEN
      RAISE eSecondsOverflow;
    END IF;
    RETURN vSeconds;
    EXCEPTION
    WHEN eSecondsOverflow THEN
      DBMS_Output.Put_Line('Error converting seconds, number out of
range:  '||
               To_Char(cSecondsPerMinute));
      RETURN 0.0;
  END Convert;

  FUNCTION Convert(pSeconds NUMBER) RETURN tTime24 IS
    vTimeBuffer tTime24;  -- return value
    vSecondsBuffer NUMBER := pSeconds;
 BEGIN
   IF pSeconds > cSecondsPerDay THEN
     RAISE eSecondsOverflow;
   END IF;
   vTimeBuffer.vHour := FLOOR(vSecondsBuffer / cSecondsPerHour);
   vSecondsBuffer := MOD(vSecondsBuffer, cSecondsPerHour);
   vTimeBuffer.vMinute := FLOOR(vSecondsBuffer / cSecondsPerMinute);
   vSecondsBuffer := MOD(vSecondsBuffer, cSecondsPerMinute);
   vTimeBuffer.vSecond := vSecondsBuffer;
   Validate(vTimeBuffer);
```

China-pub.com 下载

```
    return vTimeBuffer;
  EXCEPTION
    WHEN eSecondsOverflow THEN
      DBMS_Output.Put_Line('Error converting seconds, number too large:
'||
              To_Char(cSecondsPerMinute));
      SetTime(vTimeBuffer, 0, 0, 0);
      RETURN vTimeBuffer;
  END Convert;

  FUNCTION Add(pTime tTime24,
               pInterval TimeIntervalPackage.tTimeInterval)
    RETURN tTime24 IS
    vTimeBuffer   tTime24;  -- Buffer for return value
    vCarry  NATURAL := 0;  -- carry digits to next component
  BEGIN
    Validate(pTime);
    TimeIntervalPackage.Validate(pInterval);
    vTimeBuffer.vSecond := MOD(pTime.vSecond + pInterval.vSecond,
                               cSecondModulus);
    vCarry := FLOOR(pTime.vSecond + pInterval.vSecond /
cSecondModulus);
    vTimeBuffer.vMinute := MOD(pTime.vMinute + pInterval.vMinute +
vCarry,
                               cMinuteModulus);
    vCarry := FLOOR(pTime.vMinute + pInterval.vMinute + vCarry /
                    cMinuteModulus);
    vTimeBuffer.vHour := MOD(pTime.vHour + pInterval.vhour + vCarry,
                             cHourModulus);
    Validate(vTimeBuffer);
    RETURN vTimeBuffer;
  END Add;

  FUNCTION Add(pInterval TimeIntervalPackage.tTimeInterval,
               pTime tTime24)
    RETURN tTime24 IS
  BEGIN
    RETURN Add(pTime, pInterval);
  END Add;
FUNCTION Subtract(pTime tTime24,
                  pInterval TimeIntervalPackage.tTimeInterval)
    RETURN tTime24 IS
  vTimeBuffer tTime24;  -- Buffer for returned value
  vSeconds1 NUMBER := 0.0;
  vSeconds2 NUMBER := 0.0;
  vResultSeconds NUMBER := 0.0;
BEGIN
  -- Convert the time and interval to seconds to simplify
  vSeconds1 := Convert(pTime);
  vSeconds2 := TimeIntervalPackage.Convert(pInterval);
  vResultSeconds := vSeconds1 - vSeconds2;
  IF vResultSeconds NOT BETWEEN -cSecondsPerDay AND cSecondsPerDay
    THEN RAISE eSecondsOverflow;
  ELSIF vResultSeconds < 0 THEN
    -- Add negative value to total seconds per day to get the
    -- "reverse" number of seconds
    vResultSeconds := cSecondsPerDay + vResultSeconds;
```

China-pub.com
下载


```
    END IF;
    vTimeBuffer := Convert(vResultSeconds);
    Validate(vTimeBuffer);
    RETURN vTimeBuffer;
  END Subtract;

  FUNCTION CurrentTime RETURN tTime24 IS
    vDateTime DATE := SYSDATE;
    vTime tTime24;  -- return value
  BEGIN
    vTime.vHour := To_Number(To_Char(vDateTime, 'HH24'));
    vTime.vMinute := To_Number(To_Char(vDateTime, 'MI'));
    vTime.vSecond := To_Number(To_Char(vDateTime, 'SS')); -- no
fraction
    Validate (vTime);
    return vTime;
  END CurrentTime;

  PROCEDURE SetTime(pTime IN OUT tTime24, pHour NUMBER, pMinute NUMBER,
                    pSecond NUMBER) IS
    vTimeCopy tTime24;
  BEGIN
    pTime.vHour := pHour;
    pTime.vMinute := pMinute;
    pTime.vSecond := pSecond;
    vTimeCopy := ptime;
    Validate(vTimeCopy);
  END SetTime;
  -- 1 140
   FUNCTION Format(pTime tTime24) RETURN VARCHAR2 IS
   BEGIN
     Validate(pTime);
     RETURN To_Char(pTime.vHour)||cSeparator||
            To_Char(pTime.vMinute)||cSeparator||
            To_Char(pTime.vSecond);
   END Format;

   -- This internal procedure validates the time record
   PROCEDURE Validate(pTime IN tTime24) IS
   BEGIN
     IF pTime.vHour NOT BETWEEN 0 AND 23 THEN
       RAISE eInvalidHour;
     END IF;
     IF pTime.vMinute NOT BETWEEN 0 AND 59 THEN
       RAISE eInvalidMinute;
     END IF;
     IF pTime.vSecond NOT BETWEEN 0 AND 59.0 THEN
       RAISE eInvalidSecond;
     END IF;
   END Validate;

END Time24Package;
```

注意 对程序包有某些约束。首先，当调用程序包子程序时，Oracle执行一个隐含的保存点(savepoint)。如果子程序失败并有未处理的异常，PL/SQL在调用者产生异常之前，回滚事务处理到这个保存点。第二，如果在分布式事务处理中使用程序包，不能包含任何事务处理命令(COMMIT、ROLLBACK或SAVEPOINT)。

China-pub.com 下载

12.3.2 使用程序包

通过用程序包名、圆点分隔符和元素名组成的格式来引用程序包元素：

```
DECLARE
  vTime Time24Package.tTime24;
BEGIN
  vTime := Time24Package.CurrentTime;
END;
```

注意 如果使用服务器上的存储程序包，必须要有程序包的EXECUTE权限。和子程序不同，必须把EXECUTE权限赋给用户，而不是把一个角色(role)赋给用户。另外，定义公用同义词是一个好方法，同义词将使应用能够引用程序包，不须在程序包名前加上用户名前缀。

12.3.3 内部程序包

Form Builder的几个内部程序包提供了对应用程序很好的控制。尽管这些程序包超出了本书的讨论范围，但仔细地查阅这些程序包的参考手册的内容会对在应用中任何大量的 PL/SQL编程有好处。

Form Builder提供的程序包见表 12-2。程序包显示在 Object Navigator中的Built-in Packages结点下。

表12-2 Oracle Developer Forms中的标准包

程 序 包	描 述
DDE	提供对Microsoft DDE(动态数据交换)的访问
DEBUG	提供在代码中查找问题的调试子程序
EXEC_SQL	提供在PL/SQL代码中使用的大多数DBMS_SQL包函数
FTREE	提供对分层树的操作
OLE2	提供对Microsoft OLE(对象链接和嵌入)版本2的访问
ORA_FFI	提供对外部函数的访问
ORA_NLS	提供关于语言环境的信息
ORA_PROF	提供描述代码的定时子程序
PECS	提供性能评价工具(Oracle公司不赞成使用)
STANDARD	提供PL/SQL内部过程和函数
STANDARD Extensions	提供Oracle内部过程和函数，用于扩展PL/SQL
TEXT_IO	提供文件I/O
TOOL_ENV	提供对环境变量的访问
TOOL_ERR	提供错误处理子程序，包括MESSAGE
TOOL_RES	提供对资源文件中字符串的访问
VBX	提供对处理VBX控制工具的访问
Web	提供用于显示Web文档的Show_Document子程序

Report Builder有许多与Forms相同的程序包，但是最有特点的是 SRW包。这个程序包中的组件提供了大多数对实现特定报表编程中需要的报表元素的访问。

Graphics有许多与Forms相同的程序包，另外附加了两个程序包： OG和TOOL_INT。OG包提供了数量庞大的组件，这些组件几乎可以完成对任意图表或显示元素的所有操作。TOOL_INT包提供了给其他产品传递参数的参数列表的建立工具。

China-pub.com
下载

12.4 检查库

前面有些部分讲述了PL/SQL库。库用来把PL/SQL程序聚集并存储到独立的存储模块中，以便使用起来更容易。通常，把PL/SQL代码作为数据库对象存储。由于PL/SQL作为客户端程序设计语言，所以找到一种途径把代码作为文件系统对象存储，使其能够连接到应用程序变得十分必要。

库的工作方式与所有的Oracle Developer产品一样。可以通过Oracle Developer任意一种产品或Procedure Builder建立作为独立模块的库。然后就可以把库连接到任何Oracle Developer模块(表单、菜单、报表或图形显示)使模块中的所有组件对于任意对象可用。也可以把一个库连接到另一个模块。使用库中组件不需要加上库名作为前缀。

12.4.1 库的建立

在库中建立程序单元(过程、函数、程序包说明或程序包程序体)时，要在Object Navigator中找到库模块标题，并选择它。单击Create工具建立一个新库，或者使用Open工具打开一个以前保存的库。选择库名下的Program Unit标题，单击Create工具建立一个程序单元。首先出现New Program Unit对话框，在这个对话框中可以选择要建立哪一种程序单元(过程、函数、程序包说明或程序包程序体)，如图12-1所示。

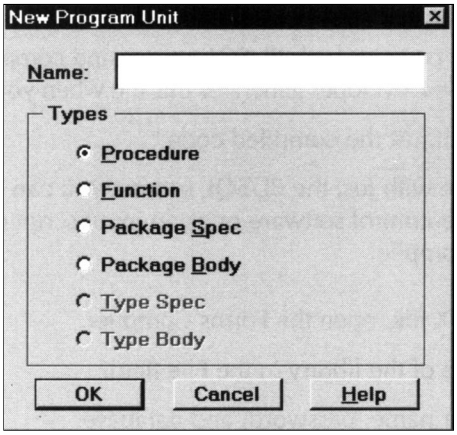


图12-1 New Program Unit对话框

选择之后单击OK按钮，出现如图12-2所示的包含选定程序单元类型模板的标准PL/SQL Editor。

代码编写结束之后，编译这个单元，通过Save工具保存库模块。保存库时也设定库的名字，不能在Navigator(导航器)中直接修改库名。确认已经对库中所有的程序单元进行了编译，如果不这样，使用库的模块会遇到问题。可以通过单元名上特殊符号存在或不存在来快速检查单元的状态：星号表示没有编译，(@)表示没有保存到磁盘。另外一个方法是在库名被选择时，选择File|Compile All菜单项，来确保编译所有的单元或看到没有编译单元的错误。

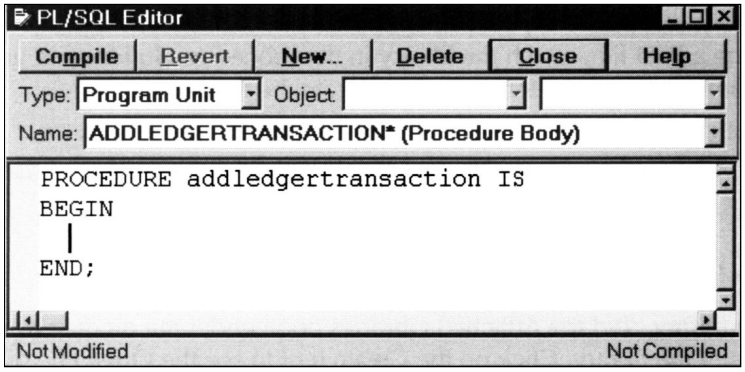


图12-2 PL/SQL Editor窗口

China-pub.com 下载

有三种类型的库文件：

- PLL：既包含PL/SQL源程序，也包含可以执行的已编译代码的文件，Oracle Developer在保存库时产生这种文件。
- PLX：只包含可以执行的已编译代码的文件。
- PLD：只包含PL/SQL源程序的文本文件，可以把这个文件作为对源控制软件的输入或能够包含和编译的PL/SQL输入脚本来使用。

1) 建立一个PLX文件，打开Forms Compiler(表单编译器)。

2) 在File域输入库的名字。

3) 输入用户名、口令和数据库。

4) 设置Module类型为FORM、MENU或LIBRARY域为LIBRARY。

5) 在“Write output to file”域输入要输出的PLX文件的文件名。图12-3举例说明了Talbot-Standard库的设置。

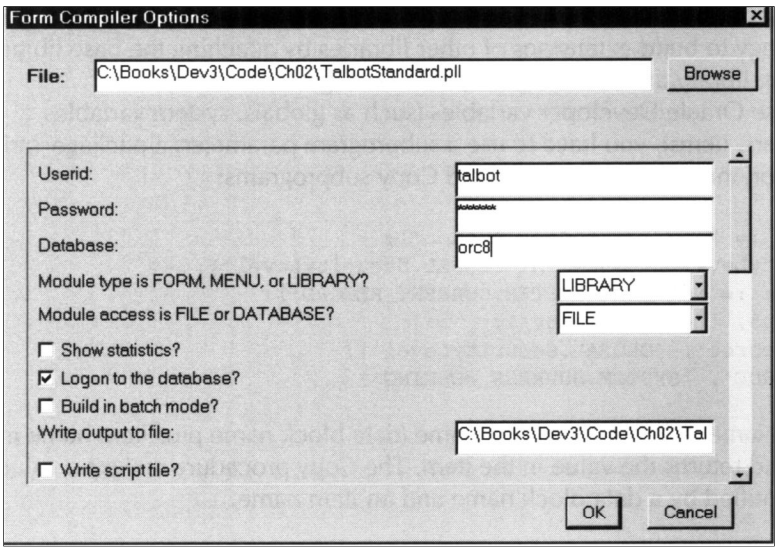


图12-3 为TalbotStandard库生成PLX文件

要建立PLD文件，在Procedure Builder中附加一个库，并在Interpreter(命令解释器)中使用EXPORT命令来生成库文本文件：

```
.EXPORT LIBRARY <library> FILE [<directory>]<name>.PLD
```

12.4.2 库的附加和使用

要附加一个库，在要附加库的模块下找到 Attached Library结点。单击Create工具，显示Open File对话框，选择要附加的库。

单击Attach按钮作为只读对象附加这个库。通过这个对象不能改变库中的任何内容。要想修改库的内容，在库模块标题下打开库进行修改。

提示 应该移除库文件名的任何路径信息。硬编码的路径意味着如果把它移到其他地方，必须改变并重新编译应用。反之，只需要改变注册变量FORMS60_PATH、REPORTS60_PATH或GRAPHICS60_PATH以包含库即可。

China-pub.com
下载

当附加一个库到另外一个模块时，模块就可以引用库的所有组件而不使用库名。在模块的Attached Library标题下库的顺序就是库名的搜索顺序。 Oracle Developer首先在Program Units部分查找程序单元。如果名字不存在，则按顺序查找每个附加库。因此可以通过把程序单元放到模块的Program Unit部分或在另外一个库前面放置一个超越库，实现特定程序单元的超越。可以使用这种方法来扩展或改编库，实现特定的用途，而不是重写或重新包装整个库。

附加一个库到另外一个库可以建立分层结构的库代码，使其他模块的使用只是简单的单击鼠标。也可以用这种方法通过附加基本库到其他库和重写程序单元的某些代码实现库的扩展。

使用Oracle Developer变量(例如全局、系统变量、参数、项)，通常要使用子程序参数、程序包变量或间接通过Name_In和Copy子程序。

```
vAmount := Name_In('Ledger.Amount');
vSecurityLevel := Name_In('GLOBAL.SecurityLevel');
vCurrent := Name_In('SYSTEM.CURRENT_RECORD');
Copy(4.05, 'Ledger.Amount');
Copy('Secret', 'GLOBAL.SecurityLevel');
Copy('TRUE', 'SYSTEM.SUPRESS_WORKING');
```

Name_In函数自变量为项名，返回值为项的值。 Copy过程把一个值连接到数据块名和项名指定的项。

注意 用子程序参数传递值比用Name_In或Copy引用变量要好。可以通过绑定变量语法(例如:Ledger.Amount)来完成。当使用Name_In/Copy方法时，把系统的两个部分通过对中间代码的直接连接联系起来，这不是一个好方法。使用子程序和程序包和参数使数据变得结构化、容易维护，库的重用性更好了。另外，如果为一个模块编写特定的代码并因此需要引用许多模块变量，应该把代码移到模块的Program Unit部分。这种方法能够直接访问变量。这些代码不能够重用，但由于设计问题，许多模块特定引用可能不能重用。

把本章学到的内容与第11章介绍PL/SQL程序设计基础结合在一起，几乎覆盖了用 Oracle Developer开发非常复杂的应用程序的所有问题。由于许多事情都能在屏幕上做，所以并不太复杂，还在于为应用程序建立一组可重用系统组件从而建立有价值的应用程序的许多方法。通过使用PL/SQL的所有特性，同时可以确保应用能够完成用户想要做的事情，并且代码将来会工作得很好，增加效率，增加给予用户的价值。

下面的两章介绍需要确定的概念，以确保组件和应用程序做想要做的事：它们都是有关测试和调试的。

China-pub.com
下载

第13章 调 试

在建立了应用程序并运行它的时候，可能会有失败。有的时候，失败还会是明显的——遗漏了块中的一个域或者在某些需要逗点的地方输入了一个“ i ”。但是多数情况遇到的失败 (错误)是不明显的，不得不去寻找它。这种查找处理或通过找到并修正代码中的错误来解决问题的技巧叫做调试。

不确切地说，调试是测试的补充。通过简单地检查代码可以找到错误，这是一个格外有效的调试途径。测试和调试结合在一起是一种强有力的技术，可以在开发过程中尽早发现并修正错误。如果开发者彻底地测试其对象并使用本章介绍的技术查找和修正运行时引起失败的错误，那么在进行alpha测试和beta测试时花的调试时间要少得多。

使用Oracle Developer来制造软件应用程序，避免了许多可能在标准程序设计语言中遇到的错误源。不存在指针和内存分配的错误，甚至 SQL语法错误也不会发生。因为用户的 SQL语句立刻被分析，并且马上就能得到错误反馈。这意味着，如果通过对另外一种程序设计语言的学习进行调试，所学的与Oracle Developer的应用会没有关系。其实， Oracle Developer也使用了相当多的优秀的调试技巧。

13.1 调试技巧

建立测试模型是彻底了解一段代码、要求或设计组件的极好的方法。正如在本章要看到的，了解一段代码也是在调试时要做的根本问题。如果为组件对象建立对象模型，通常会发现隐藏在组件中的大部分错误。实际上，既不用运行测试和不用开发测试示例也可以发现它们。建立测试模型的技巧通常对于一个好的开发者找出在对象测试中的逻辑和代码缺陷是足够了。

然而，有些错误需要进一步调查。这种情况经常发生，因为有些错误包含容易被忽略的问题，某些问题在不止一处或一个对象中发生，某些问题建立在隐含假设的基础上，而不是开发的测试模型的一部分。对于这些问题，才是调试技巧真正发挥作用的时候。

本部分的标题来自一本可能是现有的最好的关于测试和调试软件的书： Glenford Myers的《The Art of Software Testing》(Wiley,1979)。尽管这本书比较早，并且逐渐在测试和调试的技术和理论方面有些过时，但这本书对其主题提供了非常实用的介绍。没有哪个从事软件开发的人在开发软件时不读这本书的。本部分总结了 Myers介绍的调试技巧。还介绍了如何把Oracle Developer调试工具添加到调试板中和如何使用它们有效地支持调试。

13.1.1 通过归纳法定位错误

归纳的过程是从局部到整体的推理 (见图13-1)。那就是，利用故障的数据，发现故障的本质。

1) 定位相关故障数据 不要只是说“ 那里有问题 ”就开始修正。要通过留心观察精确地确定故障如何发生；如何使其再次发生；如何在相似但不同的测试情况下避免问题。要研究

China-pub.com
下载

故障发生的来龙去脉。

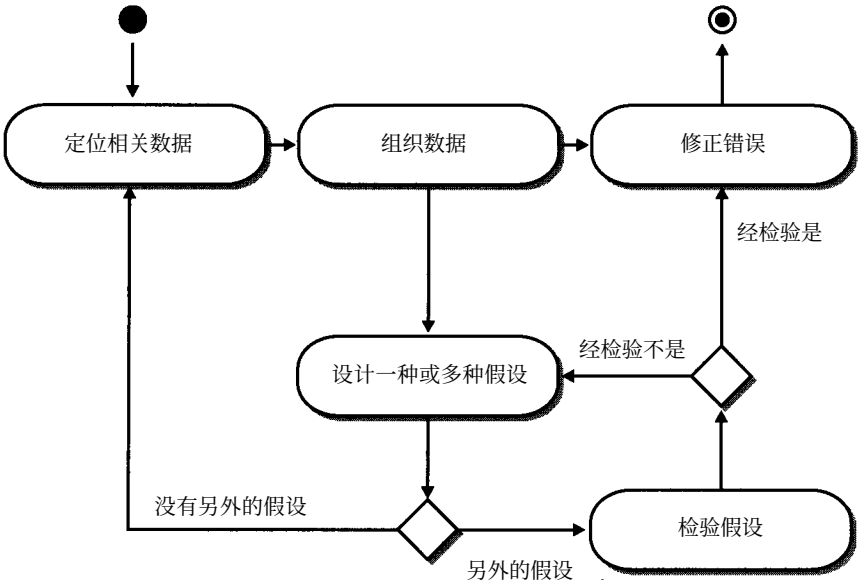


图13-1 归纳调试过程

2) 组织故障数据 通过模型分析进行归纳。如果数据组织得好，看出数据中的模型和关系就容易得多。Myers用一个表来显示发生了什么，在哪里发生，什么时候发生，以及在发生和不发生的环境下在哪个范围发生。把事件按剧本组织也十分有用。因此，要研究数据显露的关系模型。一个尤其有效的方法是回溯法：从失败的位置开始，按照进程反方向通过数据处理。另外一种可行的方法是用一些其他的测试情况进行处理，把模型看得更清楚。

3) 设计一种假设 研究数据的关系之后，对错误提出一种或多种假设。如果还不能提出假设，则需要获取更多的数据。

4) 检验假设 选择最可能的假设，然后通过它的表现检验其是否正确。首先检验它是否符合已知数据，第二，检验它在代码中如何起作用。

5) 修正错误 按照假设的建议修改代码，然后再测试代码，检验并修正那些确认了的错误，并且确认没有引起另外的错误。

当有一组数据并且相对来说不太了解程序如何工作，不太了解错误的根源时，可以使用归纳法。归纳法与后面要介绍的演绎法相比，需要进行彻底的数据分析，花大量时间和努力。

13.1.2 通过演绎法定位错误

演绎的过程是从整体到局部的推理 (见图13-2)。那就是利用对应用程序和系统的了解，通过排除和提练假设来推理错误的根源。

1) 列出可能引起失败的错误 最好与其他开发者集体讨论引起失败的错误原因。实际上，部分假设是很有用的。有的假设来自常规程序设计推测，例如查看最近改变的代码 (“我只改变了一个地方，它不可能引起故障！”)；非常复杂的代码(“我必须这样做，否则无法实现其功能！”)；或者曾经有错误的代码(“不可能再有错误了！”)。有的假设来自程序设计经验，例如边界条件错误或对SQL结构的误解，例如GROUP BY HAVING及其他外部连接(“哪一个

China-pub.com 下载

使用加号？”)。

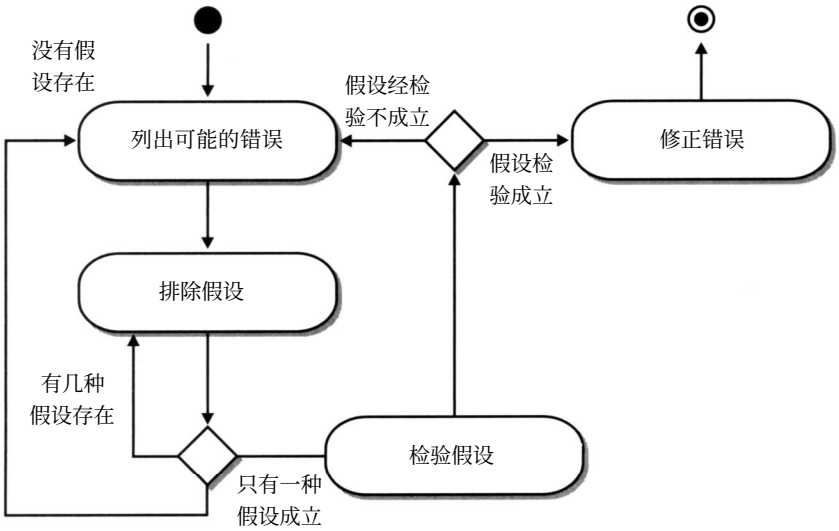


图13-2 演绎调试过程

2) 排除假设 分析数据及其关系。如果只能排除一种假设，那么着眼于剩下的假设搜集更多的数据。通常，采用某些附加测试条件会做得更好。如果排除了所有的假设，再集体讨论新的假设。

3) 精练并检验假设 使用可利用的数据尽可能地使错误推测具体化。到底要发生什么？查看所提练的假设是否与已知数据相符合，它在代码中如何起作用。

4) 修正错误 按照假设的建议修改代码，然后再测试，检验修正是否防止了错误的发生，并且没有引起另外的失败。

如果很充分了解程序的作用及失败的根源时，应该使用演绎法。这种充分了解使你能够提出一系列假设。这种方法的关键部分是通过数据分析排除并提练假设。跳过数据分析通常会导致无目的的乱撞(删去一句幽默比喻 —— 译者注)。应该检查数据，然后检验假设。

13.1.3 通过跟踪定位错误

发现错误的强制方法是跟踪程序，一个事件接一个事件，直到发现某处有错误。在调试(也就是跟踪)时，大多数开发者对要找什么有良好的直觉。但是，应该清楚这种方法比归纳法或演绎法的效率要低得多，因为在处理过程中没有“思考”。在处理过程中会产生大量的数据，其中多数是与错误不相干的。并且程序越大，越是这样。而技巧不能够递增适应性。

当然，如果不知道在程序中发生了什么并且对错误的了解不充分，那么跟踪是找到错误的唯一途径。对于Oracle Developer的程序来说，只有在不了解Oracle Developer如何处理时，这种情况才会发生。例如，不了解第五章介绍的处理过程，并且即使通过归纳处理也不能可靠地重演失败，那么就可能要通过调试程序进行单步调试，直到有异常的事情发生。这要花费一定的时间。如果没有什么发生，可能会认为Oracle Developer使用演绎法来代替强制跟踪的方法更好。

13.1.4 错误定位技术

下面是归纳法和演绎法的一些思想，没有特别的顺序：

China-pub.com
下载

- 使用大脑　这可能是对程序设计最有益的东西。
- 使用下意识　把错误放在一边，去作其他的的事。通常会灵机一动来解决你的忧虑。
- 使用朋友　和朋友讨论问题。讨论可能会使你找到一种新的假设，或者有人能够提出一种不同的假设。
- 避免强行修改　尤其尽量不要随便地改变应用程序来看要发生的事情。这比跟踪更坏，因为它分裂了执行环境，潜在地增加了新的故障。

13.1.5 修正错误

修正错误还有一些其他的处理，不仅仅是改变代码。

- 在修正错误时，注意周围的代码。
- 要确认修正了全部错误。意思是需要了解全部的问题是什么。换句话说，在没有彻底弄明白应用程序如何失败和由失败的衍生错误之前，不要开始修正错误。归纳法和演绎法调试是最有可能使你了解全部失败的方法。另外，不要只修正了表面错误而忽略了根源。软件故障比人为的故障容易消除。
- 确信在修正错误时没有破坏其他部分。尤其在应用程序是如此庞大和复杂时，每次修改可能引入新错误的概率很高。避免这种问题的最好方法是回归测试，测试能够证明所有的部分在做它应该做的事。另外一种方法是检查代码。用这种方法，需要检查所有与被修改代码相关的代码，甚至是不直接相关的代码。
- 应该研究代码中的错误模式，从错误中吸取教训。如果代码中有某个错误需要修改，那么就修改它。如果发现某种错误好象经常在 Oracle Developer的代码中发生，可在适当的位置上放置一个标准样品来看是否能够防止它们。如果发现某些错误比其他错误发生得频繁，就把工作集中到这些错误上，来减少这些错误。

13.2 调试Oracle Developer对象

假设现在完全怀疑自动调试工具作为调试方法，本章剩余部分将介绍如何有效地使用这些工具。主要是在调试的假设-检验层上使用这些工具。

Oracle Developer的环境可以以几种不同的途径进行调试。内置的调试程序可以跟踪整个代码，并且事实上能够随意地查看任何事。表单和报表也能够使用跟踪工具进行跟踪，能够知道现在程序执行到什么地方。

13.2.1 调试程序

调试程序可以在应用中以交互的现代跟踪技术跟踪 PL/SQL：

- 在源代码中交互处理断点和触发器。
- 交互查看、处理PL/SQL和变量。
- 查看PL/SQL调用堆栈。
- 交互执行PL/SQL命令。

每种工具以不同的方式建立应用程序的调试程序。

1. Forms的调试程序

在Oracle Developer Forms的应用程序中使用调试程序，必须使用调试选项生成表单和菜

China-pub.com 下载

单可执行文件。单击Object Navigator Tool Palette上的Run Form-Debug工具，或使用Program|Run Form|Debug菜单项在Debug mode(调试方式)运行。每当生成应用程序时，Oracle Developer/Forms嵌入源代码符号到应用程序中。

要在运行应用程序时运行调试程序，或者在生成器中用Run Form Debug工具或菜单项运行，或者在调试信息编译之后，用特定的调试可执行文件IFDBG60带Forms Runtime命令行参数DEBUG = YES运行:

```
IFDBG60 workerskill talbot/george@orcl debug=YES
```

运行系统在加载表单之前显示调试程序窗口，以便设置任何调试行为(见图13-3)。

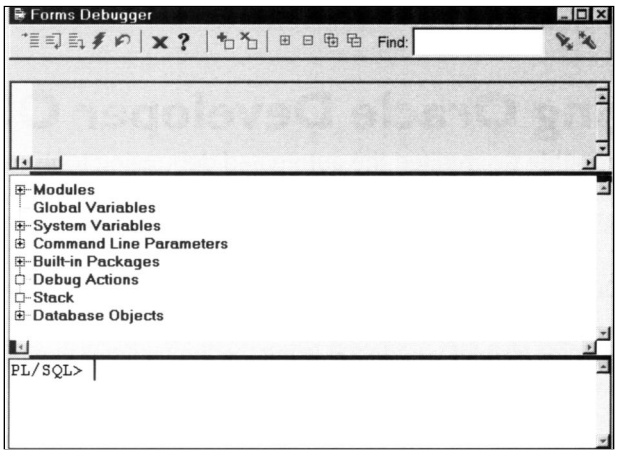


图13-3 Forms Debugger窗口

在这里，可以在任何触发器或应用的程序单元中设置断点，并且可以建立调试触发器(由调试程序事件激活的触发器)。

在顶部窗格中，调试程序显示当前执行块的PL/SQL源程序，这是源程序窗格。由于当前调试程序还没有执行一个块，所以是空的。可以在这个窗格中设置只有在调试中才存在的断点。中间窗格是Object Navigator窗格，除了少数对象，这个窗格与Object Navigator很像。这个Object Navigator有一组在运行调试时使用的对象，但它们不是在生成器的Object Navigator中的那些。使用这个窗格来查看调试行为和变量(全局、系统、局部)和调用堆栈中的参数。最下层的窗格是Interpreter(注释器)窗格，输入执行的命令在这里显示。注释器窗格用于输入命令并执行命令。

注意 如果在Forms Runtime中运行到一个好象要出错但又看不出错误和异常的地方，可以检查表单中的On-Error触发器。如果触发器用WHEN OTHERS子句捕捉了所有异常，而不是只处理特定异常并把其他的异常传递的话，那么触发器就吸收了所有的错误，而不给出任何反馈。通过注释WHEN OTHERS子句来消除这样的子句。通过插入适当的异常处理来修正问题(异常处理参见第13章)。

2. Reports调试程序

与Forms不同，Reports的调试程序把其调试行为存储在报表本身中。在Report Builder的Object Navigator中,可以看到两个标题：Debug Actions和Stack。生成器存放断点和触发器行为，后者描述运行时调用的堆栈。

China-pub.com
下载

要建立断点，在选择Report Builder Object Navigator中的程序单元后通过Program|PL/SQL Interpreter(注释器)菜单项调用Report调试程序。显示一个非模态的两个窗格 (Source程序和Interpreter窗格)的调试程序(见图13-4)。

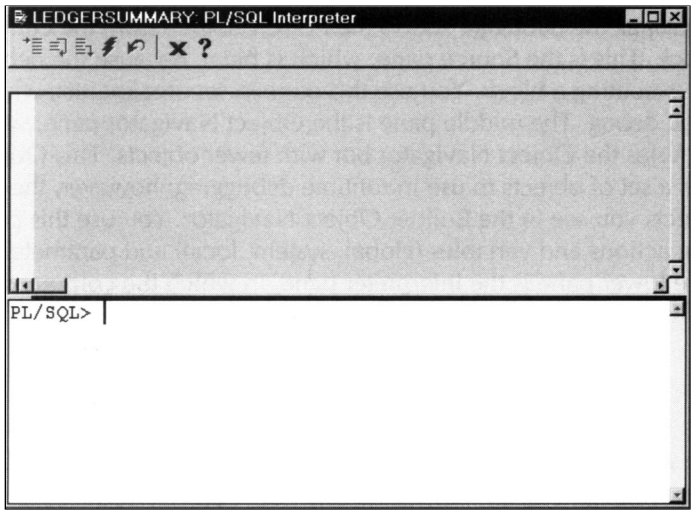


图13-4 两个窗格的调试程序

当在Object Navigator中单击一个程序单元或触发器时， PL/SQL注释器显示PL/SQL代码。然后就可以在代码的任何地方设置断点 (在下面)，然后保存报表。当运行报表时，在到达断点时，运行时调试程序弹出并高亮度显示断点 (不需要滚动窗口)。

注意 如果重新编译程序单元，需要重设受影响的断点到其新的行号。

3. Graphic调试程序

Oracle Developer Graphics调试程序的工作方式与在Oracle Developer Reports中的十分相似。在Graphics Builder中通过Tools|PL/SQL Interpreter(注释器)菜单项调用非模态的调试程序。

大多数的调试工作是找出在图形显示和图表中的哪个选项是错误的，调试程序对这些并没有很大用处。可以通过单独运行图形显示来“测试”不同的选项。到需要把图形形式集成到表单或报表中的时候，应该对图形的内部一致性有把握。

如果图形显示有运行时的错误，可以采用和表单及报表类似的调试程序。可以设置断点或调试触发器和查看变量和参数。除非在图形显示中广泛地使用 PL/SQL，否则调试程序的作用不大。

注意 如果在调试程序中修正程序错误并重新编译程序，应该在完成调试后选择Edit|Synchronize Program Units菜单项。这样就同步修改了在调试程序中的代码与在Designer或Builder中的代码。

4. Sources窗格和断点

Sources窗格以只读文本的形式显示源程序，使用源程序窗格来设置断点和查看执行过程中处于什么地方。如图13-5所示是一个在子程序体中设置有断点的调试过程。

注意窗格左边行号处的B(01)。这显示子程序中这一行上有一个断点(断点1)。可以在Sources

China-pub.com 下载

窗格中单击行来设置断点。箭头(指向B)显示调试程序在这一行前停止了程序执行。

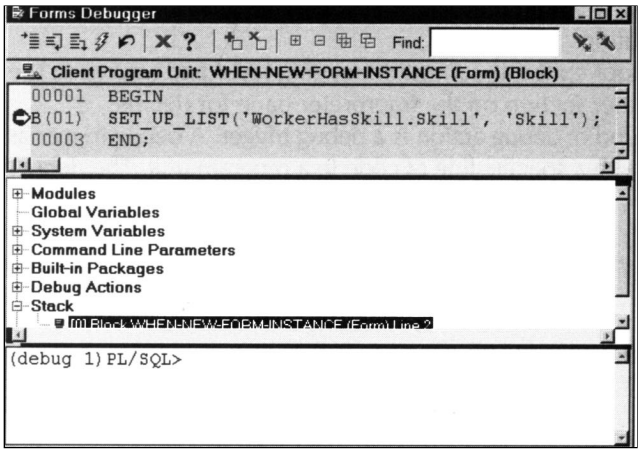


图13-5 在子程序体中设置有断点的调试过程

在最初的调试程序(比如Forms Runtime)启动时，可以在想要开始调试的地方设置一个断点。然后单击Close工具，开始执行表单。表单执行到断点处停止。当表单中断时，调试程序窗口再次显示。然后可以在Navigator窗格中检查变量(参见下一节)，或者使用Step命令开始单步执行代码。

单击Step Into工具，可以执行下一条语句。如果该语句是一个子程序调用，则进入子程序并停在子程序的第一个可执行的行上。Step Over与Step Into类似，但Step Over执行子程序，并停在子程序调用后的下一个可执行的行上。Step Out Of工具连续执行，直到当前子程序或块结束返回为止。然后调试程序停在下一个可执行的行上(见图13-6)。

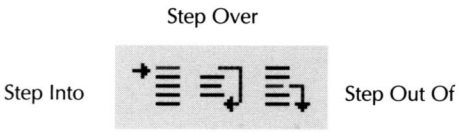


图13-6 使用Step命令

通过使用这些命令，可以在子程序中按照执行路线跟踪执行，一条语句接一条语句执行。在任何地方，都可以检查任意变量的值，查看发生了什么。

使用Go工具可以连续执行，直到遇到下一个断点或退出应用程序为止。

Reset工具退出当前的层次调试程序并返回上一个层次，参见后面关于注释器窗格的部分。

另一种调试行为是调试触发器。调试触发器把一个 PL/SQL块与程序单元代码中的一个特定行连接到一起。当执行到这行时，调试程序触发触发器。也可以设置为调试程序取得控制权或在每一个PL/SQL源程序行触发触发器。通过产生作为触发器一部分的 DEBUG.BREAK异常，可以引起中断。这样就可以设置条件断点：

```
IF DEBUG.GETN('RELDEF') = TRUE THEN
  RAISE DEBUG.BREAK;
END IF;
```

在触发器代码中，使用GETN和SETN子程序访问在当前作用域的变量。Break异常引起调试程序中断并显示当前源程序行(触发调试触发器的那行)。

警告 也可以在PL/SQL代码中用Break过程建立作用持续的断点。本书不推荐调试触发器和作用持续的断点，因为至少有两个原因。第一，主要是使用中断和调试程序检验假设，而不是广泛地跟踪代码。使用跟踪设备就可以实现上述目的。第二，如果忘记了代码中有中断并把中断遗留在代码中，用户可能会意外地进入调试程序。这不利于

China-pub.com
下载

提供一个高质量的应用程序，相反，这实际上是在代码中加入了错误。调试触发器通过自动测试条件可以帮助加速处理过程，尤其是在循环中，但是很少会用到它。

5. Navigator窗格和变量

Navigator窗格可以检查不同类型的变量。可以在Module(模块)的标题下找到任何项并查看该项。通过在相应标题下查找，可以查看全局变量、系统变量和命令参数。Stack标题提供了PL/SQL调用堆栈的当前状态。这个堆栈显示了当前执行的 PL/SQL块的分层结构。这个堆栈标题显示调试程序正在执行一个触发器 —— ON-POPULATE-DETAILS，接下来应该调用一个子程序 —— QUERY_MASTER_DETAILS(见图13-7)。

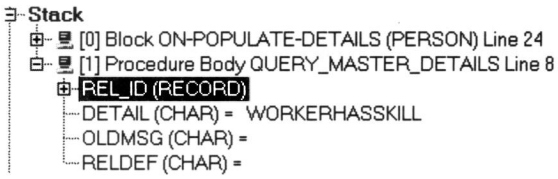


图13-7 Stack标题

在这个堆栈中，每个项代表一个正在执行的定义变量的作用域，并且可以在Stack项下找到这些变量(参数和局部变量)。如图13-8所示的例子显示在运行的 Set_Up_List子程序中的变量，包括变量ERRFLAG，被对Populate_Group的调用分配。

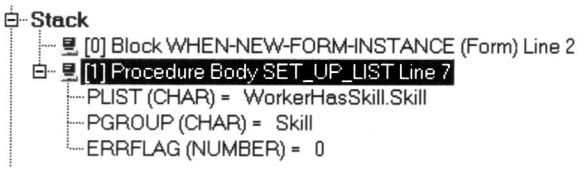


图13-8 运行Set_UP_List子程序中的变量

Navigator窗格显示所有变量的当前值，包括参数。例如， PLIST的值为WorkerHasSkill.Skill，是列表项的数据块和项名。可以通过选中并在等号后直接输入值在 Navigator中改变这些变量的值。

注意 调试程序现在不支持程序包的程序体的变量，所以没有办法查看这些变量的值。程序包的程序体变量是在程序包的程序体的说明部分说明的变量。在程序包的外部无法访问这些变量，但是在程序包中所有的子程序体和程序包初始化代码可以共享这些变量。细节参见第12章。

也可以通过断点和触发器对话框查看调试行为的细节。在 Debug Action标题下，可以选中任意的断点或调试触发器，然后双击其图标，就可以看到对话框。图 13-9是前面所述的断点的Breakpoint(断点)对话框。

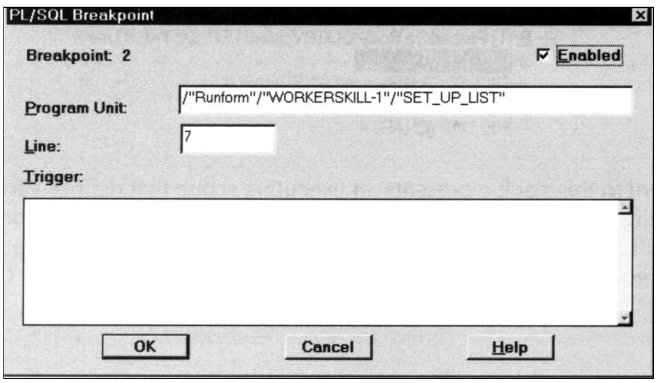


图13-9 PL/SQL Breakpoint(断点)对话框

China-pub.com 下载

使用这个对话框，可以通过Enabled复选框打开或关闭断点。也可以输入一些PL/SQL代码，作为中断发生时的触发器来执行。

6. 注释器窗格和命令

注释器窗格可以输入文本形式的调试程序命令，例如 Step Into或Go。也可以看到调试层次。调试层次是在程序执行中中断的嵌套层数。也就是说，中断程序单步执行越过子程序(Step Over)，在子程序中又遇到了中断。然后在注释器窗格中建立调试层 2。在注释窗格中可以看到一个加上圆括号的表达式作为命令提示符，例如“ (debug1)PL/SQL> ”。这个调试层的嵌套使你能够重置调试程序回到上一调试层，重新启动调试任务程序。

13.2.2 跟踪表单

在Oracle Developer Forms组件中有两个跟踪工具 —— 一个新的和一个旧的。首先说旧的。可以在运行应用程序时打开专门的调试信息。当一个事件引发一个触发器执行时，运行系统在信息行显示一个信息和一个警告窗口。单击警告窗口中的 OK按钮，应用程序继续执行。

调试信息显示触发器名和拥有触发器的对象 (表单、块或项)名(见图13-10)。

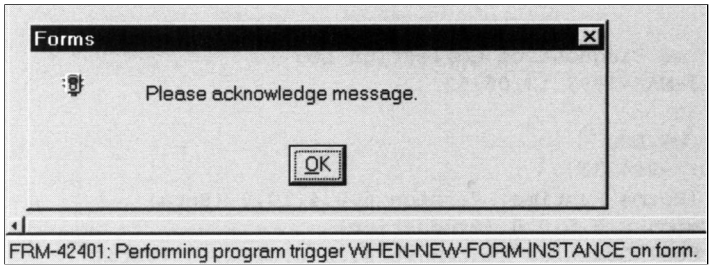


图13-10 警告窗口

要在Forms Runtime中显示信息，在Forms Runtime命令行中加上DEBUG_MESSAGES参数即可。

```
IFRUN60 workerskill talbot/george@orcl debug_messages=YES
```

要在Form Builder中运行表单时显示同样的信息，选择 Tools|Preferences菜单项并在Preferences对话框的Runtime标签上选择Debug Messages preference复选框即可。

注意 如果使用调试信息，将发现需要大量数据和很长的调试时间。可以至少尝试一次，查看触发器数量适当的表单到底发生什么。这给“ 最后一着 ”这个名词一个新的含义。

现在介绍新的跟踪工具：Forms Runtime Diagnostics(FRD，表单运行诊断)。FRD工具提供了对Oracle Developer/Forms执行的完全的跟踪，跟踪结果放在一个跟踪文件里，在你空闲时可以验证它。可能你需要大量的空闲，因为跟踪真的是很完全。

运行要跟踪的表单并在命令行中加上 RECORD和LOG参数，即可启动FRD：

```
IFRUN60 workerskill talbot/george@orcl RECORD=COLLECT LOG=WS.LOG
```

这个命令行运行表单，跟踪表单中所有的活动，并把跟踪结果写到当前工作目录下的WSP500LOG文件中。下面是这个文件前几个跟踪部分的例子：

```
Forms Runtime Diagnostics Collection Log
Created: 13-MAR-1999 12:06:52

File Name: WS.LOG
```

China-pub.com
下载


```
Process ID: -946733
Forms 6.0 (Forms Runtime) Version 6.0.4.10.0 (Beta)
PL/SQL Version 8.0.5.0.0 (Production)
Oracle Virtual Graphics System Version 6.0.5.3.0 (Beta)
Oracle Multimedia Version 6.0.2.0.0 (Production)
Oracle Tools Integration Version 6.0.5.1.0 (Production)
Oracle Tools Common Area Version 6.0.5.3.0
Oracle CORE Version 4.0.5.0.0 - Production


Opened file: C:\BOOKS\DEV3\CODE\workerskill.fmx

WHEN-NEW-FORM-INSTANCE Trigger Fired:
Form: WORKERSKILL

State Delta:
FORM WORKERSKILL
  CURBLOCK    PERSON
  CURFIELD    NAME
  STATUS      NEW
  BLOCK PERSON
    RECCOUNT    1
    TOPREC      1
    CURREC      1
    STATUS      NEW
    RECSTATUS   NEW
    FIELD NAME
      CANVAS      CANVAS2
      GEOMETRY    499,67:1694,240
      MANDATORY   TRUE
      ENABLED     TRUE
      NAVIGABLE   TRUE
      INSERTABLE  TRUE
      QUERYABLE   TRUE
      UPDATEABLE  TRUE
  BLOCK WORKERHASSKILL
    RECCOUNT    1
    TOPREC      1
    CURREC      1
    STATUS      NEW
    RECSTATUS   NEW
    FIELD NAME
      MANDATORY   TRUE
      INSERTABLE  TRUE
      UPDATEABLE  TRUE
    FIELD SKILL
      CANVAS      CANVAS2
      GEOMETRY    0,544:1000,240
      MANDATORY   TRUE
      ENABLED     TRUE
      NAVIGABLE   TRUE
      INSERTABLE  TRUE
      QUERYABLE   TRUE
      UPDATEABLE  TRUE
    FIELD ABILITY
      CANVAS      CANVAS2
      GEOMETRY    1000,544:1069,240
```

China-pub.com 下载

```
      ENABLED     TRUE
      NAVIGABLE   TRUE
      INSERTABLE  TRUE
      QUERYABLE   TRUE
      UPDATEABLE  TRUE

Executing FIND_GROUP Built-in:
In Argument 0 - Type: String    Value: Skill
Out Argument 0 - Type: Integer    Value: 1

Executing POPULATE_GROUP Built-in:
In Argument 0 - Type: Integer   Value: 1
Out Argument 0 - Type: Number    Value: 0

Executing FIND_ITEM Built-in:
In Argument 0 - Type: String    Value: WorkerHasSkill.Skill
Out Argument 0 - Type: Integer    Value: 131075

Executing Clear_List Built-in:
In Argument 0 - Type: Integer   Value: 131075

Executing FIND_ITEM Built-in:
In Argument 0 - Type: String    Value: WorkerHasSkill.Skill
Out Argument 0 - Type: Integer    Value: 131075

Executing FIND_GROUP Built-in:
In Argument 0 - Type: String    Value: Skill
Out Argument 0 - Type: Integer    Value: 1

Executing Populate_List Built-in:
In Argument 0 - Type: Integer   Value: 131075
In Argument 1 - Type: Integer   Value: 1

# 1 - WORKERSKILL:PERSON.NAME - WINDOW
END

# 1 - WORKERSKILL:PERSON.NAME
WINDOW WORKERSKILL WINDOW1 ACTIVATE 1

# 2 - WORKERSKILL:PERSON.NAME - MENU
END

# 2 - WORKERSKILL:PERSON.NAME
MENU DEFAULT Query eXecute

ON-CLEAR-DETAILS Trigger Fired:
Form: WORKERSKILL

State Delta:

Executing FIND_BLOCK Built-in:
In Argument 0 - Type: String    Value: PERSON
Out Argument 0 - Type: Integer    Value: 1

Executing GET_BLOCK_PROPERTY Built-in:
In Argument 0 - Type: Integer   Value: 1
In Argument 1 - Type: Number   Value: 160
```

China-pub.com
下载


```
Out Argument 0 - Type: String   Value: PERSON_WORKERHASSKILL

Executing FIND_RELATION Built-in:
In Argument 0 - Type: String   Value: PERSON_WORKERHASSKILL
Out Argument 0 - Type: Integer   Value: 65537

Executing GET_RELATION_PROPERTY Built-in:
In Argument 0 - Type: Integer  Value: 65537
In Argument 1 - Type: Number  Value: 152
Out Argument 0 - Type: String   Value: WORKERHASSKILL

Executing FIND_BLOCK Built-in:
In Argument 0 - Type: String   Value: WORKERHASSKILL
Out Argument 0 - Type: Integer   Value: 2

Executing GET_BLOCK_PROPERTY Built-in:
In Argument 0 - Type: Integer  Value: 2
In Argument 1 - Type: Number  Value: 100
Out Argument 0 - Type: String   Value: NEW

Executing FIND_RELATION Built-in:
In Argument 0 - Type: String   Value: PERSON_WORKERHASSKILL
Out Argument 0 - Type: Integer   Value: 65537

Executing GET_RELATION_PROPERTY Built-in:
In Argument 0 - Type: Integer  Value: 65537
In Argument 1 - Type: Number  Value: 153
Out Argument 0 - Type: String   Value: NULL

ON-POPULATE-DETAILS Trigger Fired:
Form: WORKERSKILL
Block: PERSON

State Delta:
FORM WORKERSKILL
  STATUS     QUERY
  BLOCK PERSON
    STATUS     QUERY
    RECSTATUS  QUERY
    FIELD ROWID
      VALUE      0004B006.0000.0001
    FIELD NAME
      VALUE      "Bart Sarjeant"
```

通过这个例子可以看到，跟踪给出了想要知道的详细情况，可以了解 Oracle Developer作了什么。跟踪文件给出了每个触发器的触发的和能够引起数据改变的每个事件。数据的改变伴随着一个state delta —— 显示自最近的state delta以来任何改变了的数据元素。

13.2.3 跟踪报表

用Tools|Trace菜单项打开报表跟踪，即可看到Runtime Trace Setting对话框(见图10-11)。

在这里可以复选想要跟踪日志显示的统计信息。下面是在复选了所有跟踪选项产生的Ledger Summary报表跟踪日志的第一部分：

```
LOG :
      Report: Ledger
      Logged onto server:
      --
```

China-pub.com 下载

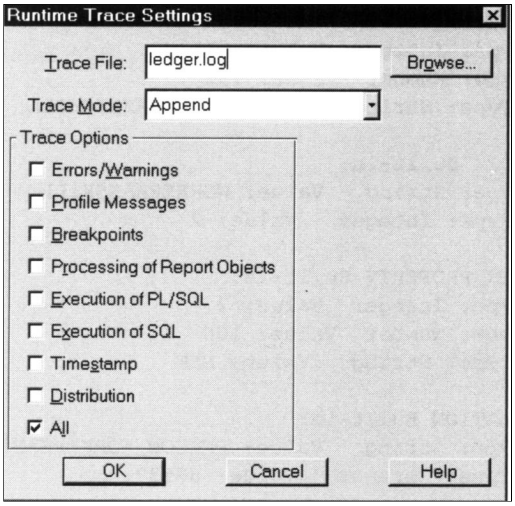


图13-11 Runtime Trace Setting对话框

```
      Username:


LOG :
      Logged onto server: orcl
      Username: talbot


15:33:48  APP  (  Header
15:33:48  APP . (  Chart                    BOUGHTDISPLAY
15:33:53  APP . )  Chart                    BOUGHTDISPLAY
15:33:53  APP  )  Header
15:33:56  APP  (  Frame
15:33:56  APP . (  Generic Graphical Object B_1
15:33:56  APP . )  Generic Graphical Object B_1
15:33:56  APP . (  Text Boilerplate         B_DATE_BLULOGO1
15:33:57  APP . )  Text Boilerplate         B_DATE_BLULOGO1
15:33:57  APP . (  Text Boilerplate         B_TITLE_BLULOGO1
15:33:57  APP . )  Text Boilerplate         B_TITLE_BLULOGO1
15:33:57  APP . (  Text Boilerplate         OR$_BPAGENUM_BLULOGO1
15:33:57  APP . )  Text Boilerplate         OR$_BPAGENUM_BLULOGO1
15:33:57  APP . (  Text Field               F_DATE1
15:33:57  APP .. (  Database Column          Name unknown
15:33:57  APP .. )  Database Column          Name unknown
15:33:57  APP . )  Text Field               F_DATE1
15:33:57  APP  )  Frame
15:33:57  APP  (  Frame
15:33:57  APP . (  Frame                    M_G_PERSON_GRPFR
15:33:57  APP .. (  Repeating Frame          R_G_PERSON
15:33:57  APP ... (  Group                    G_Person  Local Break:  0  Global Break:  0
15:33:57  APP .... (  Query                    LedgerQuery
15:33:57  SQL         EXECUTE QUERY : select Person , ActionDate , Item , Quantity , Rate
                      from Ledger  ORDER BY 1 ASC , 1 , 2
15:33:57  APP .... )  Query                    LedgerQuery
15:33:57  PLS .... (  Function:       amountformula
15:33:57  PLS .... )  Function:       amountformula
15:33:57  APP ... )  Group                    G_Person
15:33:57  APP ... (  Text Field               F_PERSON
15:33:57  APP .... (  Database Column          Person
15:33:57  APP .... )  Database Column          Person
15:33:57  APP ... )  Text Field               F_PERSON
15:33:57  APP ... (  Frame                    M_G_ACTIONDATE_GRPFR
15:33:57  APP .... (  Repeating Frame          R_G_ACTIONDATE
15:33:57  APP ..... (  Group                    G_ActionDate  Local Break:  0  Global Break:  0
15:33:57  APP ..... )  Group                    G_ActionDate
15:33:57  APP ..... (  Text Field               F_RATE
15:33:57  APP ..... (  Database Column          Rate
15:33:57  APP ..... )  Database Column          Rate
15:33:57  APP ..... )  Text Field               F_RATE
15:33:57  APP ..... (  Text Field               F_QUANTITY
15:33:57  APP ..... (  Database Column          Quantity
15:33:57  APP ..... )  Database Column          Quantity
15:33:57  APP ..... )  Text Field               F_QUANTITY
```

China-pub.com
下载


```
15:33:57  APP ..... (  Text Field              F_ITEM
15:33:57  APP ...... (  Database Column         Item
15:33:57  APP ...... )  Database Column         Item
15:33:57  APP ..... )  Text Field              F_ITEM
15:33:57  APP ..... (  Text Field              F_ACTIONDATE
15:33:57  APP ...... (  Database Column         ActionDate
15:33:57  APP ...... )  Database Column         ActionDate
15:33:57  APP ..... )  Text Field              F_ACTIONDATE
15:33:57  APP ..... (  Group                   G_ActionDate  Local Break:  1  Global Break:  1
15:33:57  PLS ...... (  Function:       amountformula
15:33:57  PLS ...... )  Function:       amountformula
15:34:05  APP ... )  Frame                     M_G_ACTIONDATE_GRPFR
15:34:05  APP ... (  Group                     G_ActionDate  Local Break:  1  Global Break:  225
15:34:05  APP ... )  Group                     G_ActionDate
15:34:05  APP ... (  Group                     G_Person  Local Break:  52  Global Break:  52
15:34:05  APP ... )  Group                     G_Person
15:34:05  APP .. )  Repeating Frame           R_G_PERSON
15:34:05  APP .. (  Frame                     M_G_PERSON_HDR
15:34:05  APP ... (  Text Boilerplate          B_RATE
15:34:05  APP ... )  Text Boilerplate          B_RATE
15:34:05  APP ... (  Text Boilerplate          B_QUANTITY
15:34:05  APP ... )  Text Boilerplate          B_QUANTITY
15:34:05  APP ... (  Text Boilerplate          B_ITEM
15:34:05  APP ... )  Text Boilerplate          B_ITEM
15:34:05  APP ... (  Text Boilerplate          B_ACTIONDATE
15:34:05  APP ... )  Text Boilerplate          B_ACTIONDATE
15:34:05  APP ... (  Text Boilerplate          B_PERSON
15:34:05  APP ... )  Text Boilerplate          B_PERSON
15:34:05  APP .. )  Frame                     M_G_PERSON_HDR
15:34:05  APP . )  Frame                     M_G_PERSON_GRPFR
15:34:05  APP  )  Frame

                    +-------------------------------------+
                    | Report Builder Profiler statistics  |
                    +-------------------------------------+

          TOTAL ELAPSED Time:      60.00 seconds

         Reports Time:      60.00 seconds (100.00% of TOTAL)

                ORACLE Time:       0.00 seconds ( 0.00% of TOTAL)

                       UPI:        0.00 seconds
                       SQL:        0.00 seconds

TOTAL CPU Time used by process: N/A
```

与其跟踪整个报表，不如使用 SRW包通过在报表触发器代码中调用 SRW.TRACE_START()和SRW.TRACE_END()来打开对报表处理中特定处的跟踪。可以通过 SRW.TRACE_ADD_OPTIONS()和SRW.TRACE_REM_OPTIONS()来设置跟踪选项。

13.2.4 跟踪SQL和客户机/服务器事件

在某些情况下有两种跟踪可能会有用处，尽管严格地说它们并不是 Oracle Developer的一部分。通常有这种情况经常发生，就是你不知道在发生什么并且不得不从大量的几乎没用的资料去寻找发生的问题，所以客户机和服务器级的跟踪仍然是对应用程序开发者有价值的工具。再次给“最后一着”以新的含义。

1. 在客户机端的SQL*NET跟踪

如果使用 Oracle 7数据库服务器和 SQL*Net版本2从客户机连接到服务器，可以使用SQL*Net跟踪。它提供了 SQL*Net知道的所有事件的十六进制转储 (dump)。可以直接看到哪个字符串从客户机端发送到服务器，这也许足够用来确定错误。这种跟踪对于调试第三代程序很有帮助，在其中可能通过改写存储器位置或偶尔在字符串末尾连接上多余的字符串造成奇怪的情况。这在Oracle Developer中很难作到。

打开SQL*Net跟踪需要在Oracle_Home/Network/Admin目录下的SQLNET.ORA文件中设置两个参数：TRACE_LEVEL_CLIENT(值USER或ADMIN为网络信息摘要或详细数据， OFF

China-pub.com 下载

为关闭跟踪)和TRACE_DIRECTORY_CLIENT(值为要建立跟踪文件的目录)。跟踪结果写在指定目录下的SQLNET.TRC文件中。

注意 如果上面的说明不充分，向Oracle Corporation Technical Support(Oracle公司技术支持)查询使用这种技术的更多信息。这些细节相当复杂，超出了本书的范围。由于跟踪文件增长得非常快，要确定有足够的磁盘空间，并且在取得了需要得跟踪信息后及时关闭跟踪。不要在网络基础上打开跟踪，应该仅仅是在本地 (局部变量设置或SQLNET.ORA本地副本)，否则网络系统管理员会发现系统中的可用磁盘空间的急剧减少。

2. 在客户机端的Net8跟踪

如果使用Oracle 8数据库服务器，可以使用Net8从客户机连接到服务器，从而可以使用Net 8跟踪查看发送到服务器的SQL(也包括所有其他网络相关事件)。Net8提供了所知道的所有事件的十六进制转储。可以直接看到哪个字符串从客户机端发送到服务器，这也许足够用来确定错误。这种跟踪对于调试自定义的Pro*C或Oracle Call Interface(Oracle 调用接口)程序很有帮助，在其中可能通过改写存储器位置或偶尔在字符串末尾连接上多余的字符串造成奇怪的情况。这在Oracle Developer中很难作到。

打开Net8跟踪很容易。在客户机端Net8 Admin目录下找到SQLNET.ORA文件，通常是在Oracle_Home\Net80\Admin这样一些地方。通常在文件中会发现一个与跟踪相关的行，TRACE_LEVEL_CLIENT=OFF。把OFF改为SUPPORT并加上两行来给跟踪文件起名字：

```
TRACE_LEVEL_CLIENT = SUPPORT
TRACE_FILE_CLIENT = Net8_Trace
TRACE_DIRECTORY_CLIENT = C:\Orawin95\Net80\Trace
```

这个例子打开了SUPPORT-level跟踪并设置跟踪文件为C:\Orawin95\Net80\Trace\ Net8_Trace.trc。可以随便给文件起名字，但不要加扩展名。可以随便设置目录，但要确保在运行一个客户机Oracle程序之前这个目录必须存在。LEVEL有另外的级别值，包括USER和ADMIN，但它们对输出有限制。SUPPORT可能是最好的选择，因为它给出了所有存在的信息并且有一个跟踪文件格式化工具可用，如trcasst.

注意 关于使用Net8进行跟踪和跟踪辅助工具的更多信息，查阅 Oracle Network Administration(网络管理员)文档。例如，如果运行第4章的Ledger应用程序并按下Execute Query按钮来查询所有记录，将得到下面的Net8跟踪文件：

```
(3ff2) nspsend: 00 FD 00 00 06 00 00 00  |........|
(3ff2) nspsend: 00 00 03 5E 0D 61 80 00  |...^.a..|
(3ff2) nspsend: 00 00 00 00 00 98 F2 00  |........|
(3ff2) nspsend: 01 84 00 00 00 C8 9F 00  |........|
(3ff2) nspsend: 01 09 00 00 00 00 00 00  |........|
(3ff2) nspsend: 00 EC 9F 00 01 00 00 00  |........|
(3ff2) nspsend: 00 0A 00 00 00 1E 00 00  |........|
(3ff2) nspsend: 00 00 00 00 00 00 00 00  |........|
(3ff2) nspsend: 00 00 00 00 00 00 00 00  |........|
(3ff2) nspsend: 00 00 00 00 00 00 00 00  |........|
(3ff2) nspsend: 00 EE 9F 00 01 53 45 4C  |.....SEL|
(3ff2) nspsend: 45 43 54 20 52 4F 57 49  |ECT ROWI|
(3ff2) nspsend: 44 2C 4C 2E 50 45 52 53  |D,L.PERS|
(3ff2) nspsend: 4F 4E 2C 4C 2E 4C 45 44  |ON,L.LED|
```

China-pub.com
下载


```
(3ff2) nspsend: 47 45 52 49 44 2C 4C 2E  |GERID,L.|
(3ff2) nspsend: 41 43 54 49 4F 4E 44 41  |ACTIONDA|
(3ff2) nspsend: 54 45 2C 4C 2E 41 43 54  |TE,L.ACT|
(3ff2) nspsend: 49 4F 4E 2C 4C 2E 49 54  |ION,L.IT|
(3ff2) nspsend: 45 4D 2C 4C 2E 51 55 41  |EM,L.QUA|
(3ff2) nspsend: 4E 54 49 54 59 2C 4C 2E  |NTITY,L.|
(3ff2) nspsend: 51 55 41 4E 54 49 54 59  |QUANTITY|
(3ff2) nspsend: 54 59 50 45 2C 4C 2E 52  |TYPE,L.R|
(3ff2) nspsend: 41 54 45 2C 4C 2E 41 4D  |ATE,L.AM|
(3ff2) nspsend: 4F 55 4E 54 2C 4C 2E 50  |OUNT,L.P|
(3ff2) nspsend: 45 52 53 4F 4E 2E 4E 41  |ERSON.NA|
(3ff2) nspsend: 4D 45 20 46 52 4F 4D 20  |ME FROM |
(3ff2) nspsend: 4C 45 44 47 45 52 20 4C  |LEDGER L|
(3ff2) nspsend: 20 01 00 00 00 01 00 00  | .......|
(3ff2) nspsend: 00 00 00 00 00 00 00 00  |........|
(3ff2) nspsend: 00 00 00 00 00 00 00 00  |........|
(3ff2) nspsend: 00 00 00 00 00 01 00 00  |........|
(3ff2) nspsend: 00 00 00 00 00 00 00 00  |........|
```

3. 在服务器端的SQL跟踪

在服务器端，使用SQL跟踪可以跟踪SQL语句及其性能指数。

在Oracle Developer/Forms中，通过在命令行中的Statistics选项打开SQL跟踪。

```
IFRUN60 workerskill talbot/george@orcl statistics=YES
```

注意 我未能在Oracle Developer组件(6.0.5.0.2)中通过IFRUN60命令行参数启动statistics(统计信息)跟踪。Oracle公司证明它支持这个功能，可以试一试。

在Reports中，在Before-Parameter-Form触发器中通过SRW过程调用来打开跟踪：

```
SRW.Do_SQL('ALTER SESSION SET SQL_TRACE TRUE');
```

在Graphics中，在 Open触发器中使用类似的调用：

```
Do_SQL('ALTER SESSION SET SQL_TRACE TRUE');
```

注意 也可以通过设置Oracle实例的SQL_TRACE参数TRUE,为所有会话打开跟踪。通常会想这样做，但其影响范围过大。因为这样会在每个人使用服务器时跟踪每个细小的SQL处理过程。

当为应用程序打开跟踪时，服务器产生一个放在转储目录中的跟踪文件。可以通过运行sysmgr和下面的命令来确定这个目录：

```
SHOW PARAMETER USER_DUMP_DEST
```

也可以从Initialization Parameters(初始化参数)窗口中的Instance Manager(实例管理器)中取得该信息。

在Windows NT操作系统中，跟踪文件目录可能是Oracle_Home\RDBMS80\Trace或类似的目录。在目录中找到扩展名为 .TRC并且主文件名是唯一值 (进程ID或类似的数字)的文件，例如ORA00075.TRC。例如，运行Ledger应用程序，产生如下跟踪文件：

```
Dump file D:\orant\rdbms80\trace\ORA00075.TRC
Mon Mar 29 09:50:48 1999
ORACLE V8.0.5.0.0 - Production vsnsta=0
vsnsql=c vsnxtr=3
Windows NT V4.0, OS V5.101, CPU type 586
Oracle8 Enterprise Edition Release 8.0.5.0.0 - Production
With the Partitioning and Objects options
PL/SQL Release 8.0.5.0.0 - Production
```

China-pub.com 下载

```
Windows NT V4.0, OS V5.101, CPU type 586
Instance name: orc8

Redo thread mounted by this instance: 1

Oracle process number: 10

pid: 4b


*** SESSION ID:(8.37) 1999.03.29.09.50.48.306
=====================
PARSING IN CURSOR #1 len=32 dep=0 uid=25 oct=42 lid=25 tim=22639233 hv=2162858323 ad='29e4478'
ALTER SESSION SET SQL_TRACE TRUE
END OF STMT
EXEC #1:c=3,e=3,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=4,tim=22639234
=====================
PARSING IN CURSOR #1 len=32 dep=0 uid=25 oct=42 lid=25 tim=22639503 hv=2162858323 ad='29e4478'
ALTER SESSION SET SQL_TRACE TRUE
END OF STMT
PARSE #1:c=1,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22639503
EXEC #1:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22639503
=====================
PARSING IN CURSOR #2 len=132 dep=0 uid=25 oct=3 lid=25 tim=22640182 hv=1199613752 ad='2a54244'
SELECT
ROWID,L.PERSON,L.LEDGERID,L.ACTIONDATE,L.ACTION,L.ITEM,L.QUANTITY,L.QUANTITYTYPE,L.RATE,L.AMOUNT,L.P
ERSON.NAME FROM LEDGER L
END OF STMT
PARSE #2:c=1,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22640183
EXEC #2:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22640183
=====================
PARSING IN CURSOR #4 len=47 dep=1 uid=0 oct=3 lid=0 tim=22640184 hv=1158616671 ad='2b6c630'
select metadata from kopm$  where name='DB_FDO'
END OF STMT
PARSE #4:c=1,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,tim=22640184
EXEC #4:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,tim=22640184
FETCH #4:c=0,e=0,p=0,cr=2,cu=0,mis=0,r=1,dep=1,og=4,tim=22640184
STAT #4 id=1 cnt=1 pid=0 pos=0 obj=240 op='TABLE ACCESS BY INDEX ROWID KOPM$ '
STAT #4 id=2 cnt=1 pid=1 pos=1 obj=241 op='INDEX UNIQUE SCAN '
FETCH #2:c=2,e=2,p=0,cr=21,cu=3,mis=0,r=11,dep=0,og=4,tim=22640185
=====================
PARSING IN CURSOR #4 len=156 dep=0 uid=25 oct=3 lid=25 tim=22640616 hv=3424553781 ad='2a121cc'
SELECT L.PERSON,L.LEDGERID,L.ACTIONDATE,L.ACTION,L.ITEM,L.QUANTITY,L.QUANTITYTYPE,L.RATE,L.AMOUNT
FROM LEDGER L WHERE ROWID=:1 FOR UPDATE OF L.PERSON NOWAIT
END OF STMT
PARSE #4:c=1,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22640616
EXEC #4:c=0,e=0,p=0,cr=1,cu=3,mis=0,r=0,dep=0,og=4,tim=22640616
FETCH #4:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=1,dep=0,og=4,tim=22640617
=====================
PARSING IN CURSOR #1 len=14 dep=0 uid=25 oct=46 lid=25 tim=22640800 hv=278968606 ad='2a112a8'
SAVEPOINT FM_1
END OF STMT
PARSE #1:c=1,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22640800
EXEC #1:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22640800
=====================
PARSING IN CURSOR #5 len=155 dep=0 uid=25 oct=6 lid=25 tim=22640801 hv=1725894326 ad='2a0eef4'
UPDATE LEDGER L SET
L.PERSON=:1,L.LEDGERID=:2,L.ACTIONDATE=:3,L.ACTION=:4,L.ITEM=:5,L.QUANTITY=:6,L.QUANTITYTYPE=:7,L.RA
TE=:8,L.AMOUNT=:9 WHERE ROWID=:10
END OF STMT
PARSE #5:c=1,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22640802
EXEC #5:c=0,e=0,p=0,cr=0,cu=1,mis=0,r=1,dep=0,og=4,tim=22640802
XCTEND rlbk=0, rd_only=0
EXEC #4:c=1,e=1,p=0,cr=1,cu=3,mis=0,r=0,dep=0,og=4,tim=22640937
FETCH #4:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=1,dep=0,og=4,tim=22640937
=====================
PARSING IN CURSOR #1 len=14 dep=0 uid=25 oct=46 lid=25 tim=22641088 hv=278968606 ad='2a112a8'
SAVEPOINT FM_1
END OF STMT
PARSE #1:c=1,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22641088
EXEC #1:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22641088
EXEC #5:c=0,e=0,p=0,cr=0,cu=1,mis=0,r=1,dep=0,og=4,tim=22641089
XCTEND rlbk=0, rd_only=0
=====================
```

China-pub.com
下载


```
PARSING IN CURSOR #1 len=8 dep=0 uid=25 oct=45 lid=25 tim=22641236 hv=1470906206 ad='2a50020'
ROLLBACK
END OF STMT
PARSE #1:c=1,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22641237
XCTEND rlbk=1, rd_only=1
EXEC #1:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22641237
XCTEND rlbk=0, rd_only=1
STAT #2 id=1 cnt=11 pid=0 pos=0 obj=2488 op='TABLE ACCESS FULL LEDGER '
STAT #4 id=1 cnt=2 pid=0 pos=0 obj=0 op='FOR UPDATE '
STAT #4 id=2 cnt=2 pid=1 pos=1 obj=2488 op='TABLE ACCESS BY USER ROWID LEDGER '
STAT #5 id=1 cnt=0 pid=0 pos=0 obj=0 op='UPDATE LEDGER '
STAT #5 id=2 cnt=2 pid=1 pos=1 obj=2488 op='TABLE ACCESS BY USER ROWID LEDGER '
```

这个文件有些难读。可以使用 Oracle Toolkit profiler program(TKPROF,)来把这个跟踪文件格式化，使其容易使用。在 TKPROF命令行提供输入 TRC文件名和输出文件名。还要输入用户名和口令：

```
TKPROF80 c:\orant\rdbms80\trace\ora00075.trc c:\temp\output.trc
```

格式化LEDGER应用程序的跟踪文件产生下面格式的文件：

```
TKPROF: Release 8.0.5.0.0 - Production on Mon Mar 29 10:4:24 1999

(c) Copyright 1998 Oracle Corporation.  All rights reserved.

Trace file: ora00075.trc
Sort options: default

********************************************************************************
count    = number of times OCI procedure was executed
cpu      = cpu time in seconds executing
elapsed  = elapsed time in seconds executing
disk     = number of physical reads of buffers from disk
query    = number of buffers gotten for consistent read
current  = number of buffers gotten in current mode (usually for update)
rows     = number of rows processed by the fetch or execute call
********************************************************************************

ALTER SESSION SET SQL_TRACE TRUE


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0          0          0           0
Execute      2      0.03       0.03          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      0.04       0.04          0          0          0           0

Misses in library cache during parse: 0
Misses in library cache during execute: 1
Optimizer goal: CHOOSE
Parsing user id: 25
********************************************************************************

SELECT ROWID,L.PERSON,L.LEDGERID,L.ACTIONDATE,L.ACTION,L.ITEM,L.QUANTITY,
  L.QUANTITYTYPE,L.RATE,L.AMOUNT,L.PERSON.NAME
FROM
 LEDGER L


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1      0.02       0.02          0         19          3          11
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      0.03       0.03          0         19          3          11
```

China-pub.com 下载

```
Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: 25
********************************************************************************

select metadata
from
 kopm$  where name='DB_FDO'


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1      0.00       0.00          0          2          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      0.01       0.01          0          2          0           1

Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: SYS   (recursive depth: 1)
********************************************************************************

SELECT L.PERSON,L.LEDGERID,L.ACTIONDATE,L.ACTION,L.ITEM,L.QUANTITY,
  L.QUANTITYTYPE,L.RATE,L.AMOUNT
FROM
 LEDGER L WHERE ROWID=:1 FOR UPDATE OF L.PERSON NOWAIT


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0          0          0           0
Execute      2      0.01       0.01          0          2          6           0
Fetch        2      0.00       0.00          0          0          0           2
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        5      0.02       0.02          0          2          6           2

Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: 25
********************************************************************************

SAVEPOINT FM_1


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        2      0.02       0.02          0          0          0           0
Execute      2      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.02       0.02          0          0          0           0

Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: 25
********************************************************************************

UPDATE LEDGER L SET L.PERSON=:1,L.LEDGERID=:2,L.ACTIONDATE=:3,L.ACTION=:4,
  L.ITEM=:5,L.QUANTITY=:6,L.QUANTITYTYPE=:7,L.RATE=:8,L.AMOUNT=:9
WHERE
 ROWID=:10


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0          0          0           0
Execute      2      0.00       0.00          0          0          2           2
Fetch        0      0.00       0.00          0          0          0           0
```

China-pub.com
下载


```
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      0.01       0.01          0          0          2           2

Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: 25
********************************************************************************

ROLLBACK
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.01       0.01          0          0          0           0

Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: 25


********************************************************************************

OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        7      0.07       0.07          0          0          0           0
Execute     10      0.04       0.04          0          2          8           2
Fetch        3      0.02       0.02          0         19          3          13
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       20      0.13       0.13          0         21         11          15

Misses in library cache during parse: 0
Misses in library cache during execute: 1


OVERALL TOTALS FOR ALL RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1      0.00       0.00          0          2          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      0.01       0.01          0          2          0           1

Misses in library cache during parse: 0

    8  user  SQL statements in session.
    1  internal SQL statements in session.
    9  SQL statements in session.
********************************************************************************
Trace file: ora00075.trc
Trace file compatibility: 7.03.02
Sort options: default

       1  session in tracefile.
       8  user  SQL statements in trace file.
       1  internal SQL statements in trace file.
       9  SQL statements in trace file.
       7  unique SQL statements in trace file.
      89  lines in trace file.
```

注意 可以通过设置Oracle实例的TIMED_STATISTICS参数为TRUE，来打开时间统计信息(timing statistics)。如果这个参数为FALSE，在统计信息表中只能看到0。应该清楚，

China-pub.com 下载

打开这个参数会明显地降低系统的速度，所以，不要在产品系统中打开这个参数，除非你渴望让所有的用户都受到影响。另外，应该查看Enterprise Manager(企业管理器)中新的Oracle Trace选项 —— Performance Pack(性能压缩)选项的一部分， Enterprise Manager中提供了更多跟踪Oracle应用程序的工具。

第三部分介绍了如何使用 Oracle Developer的高级工具编写代码和调试应用程序。系统地使用这些工具，可以建立一个高质量的系统。通过系统中的可重用的代码，可以建立许多应用程序。下一部分将介绍如何把应用程序从开发环境中移到现实的环境中，交付客户使用并且进行有效维护。另外也将介绍现实的环境中如何把应用程序与其他软件结合在一起，如何与其他人的创造性成果和你自己的有效方法相结合。

China-pub.com
下载

第四部分　高 级 工 具

第14章　部署和配置应用程序

在完成了应用程序原型，调查了用户的意见，重新设计和确保了应用程序的安全，通过了编译、测试和调试之后，剩下要做的事情是什么呢？

在完成应用程序之后所要做的事情，是将应用程序交付给用户的过程中真正关键的问题，这也是出于自己爱好或为练习而编程与开发软件产品之间的区别。你已经在开发工具中运行了应用程序，现在是建立Application Server和在因特网上运行的时候了。

14.1　部署应用程序

部署软件产品就是将它分发给用户。它包括将软件打包并安装到用户的工作环境或网络上。

首先你必须决定部署模式：客户机 /服务器模式或三方部署模式。三方部署模式利用Application Server作为客户和数据库服务器的第三方。

- 客户机/服务器部署模式需要将一个安装程序包安装到每一个客户机上，包括 Oracle Developer客户机的运行时间引擎、可执行的应用程序和其他各种各样运行应用程序所需要的文件。
- 三方发送模式需要在一个中心服务器上安装 Oracle Developer Server驱动程序、应用程序可执行文件及相关文件、作为Oracle Application Server一部分的HTTP接收器或者其他Web服务器。

三方部署模式的好处是非常明显的，它便于更多用户使用应用程序。因为只需要安装一次，故不必制作安装系统，也不必四处奔走为所有的用户安装。此外，客户机仅需要网络浏览器或Java程序浏览器即可运行应用程序，而不需整个 Oracle Developer和与应用程序相关的软件。这可节省计算机空间去做其他事情，而且不必在客户机维护该软件，从而减少系统的维护和管理费用。

14.2　复杂的Web

可以想象一下，如果发送一个应用软件所要做的就是告诉用户一个它的名字，那该多好。这就是因特网的承诺以及 Oracle Developer提供的多层部署平台 Oracle Developer Server的能力。

Web浏览器是连接网络的一个应用程序，通常是在因特网或企业内部网上利用 HTML主页浏览文件。Sun Microsystems加上Java这种面向网络的程序设计语言，生成一种真正有效的部署软件的方法。这样就可以通过浏览 HTML网页、轻点按钮运行Java程序来使用应用程序。

Oracle Developer Server是一个工作平台，它帮助你通过 Web浏览器部署应用程序。并且

China-pub.com 下载

它使应用程序的部署变得更简单、更廉价，让你创造出更多优秀实用的 Web应用程序。Oracle Developer Server可以充分利用因特网的优点和客户 /服务器的计算功能。使用基于 Web的应用程序可以达到脱离工作平台的高度，通过浏览器可以在任何一部机器上运行你的应用程序，而不需要在不同的操作系统下重新编译。可以得到多种功能的表单和报表，包括交互式表单、HTML和Adobe Acrobat报表、下钻报表和图形、动态报表和报表队列监控。

图14-1说明了Oracle Developer Server Forms的组织结构。客户工作站运行一个可运行Java的Web浏览器。用户对HTML页面输入一个URL进入应用程序开始页面 (例如http://www.talbotfarms.com/hr)。当用户在浏览器中运行一个表单文件时，该浏览器就下载 Oracle Developer Forms Java Client小应用程序(关于用户界面说明的一个小小程序代码)并运行它。该程序向Forms Server Listener发出请求，要求服务器运行应用程序。 Listener连接到Forms Runtime Server上，并传送运行时间参数，包括用户名和数据库服务器的口令。

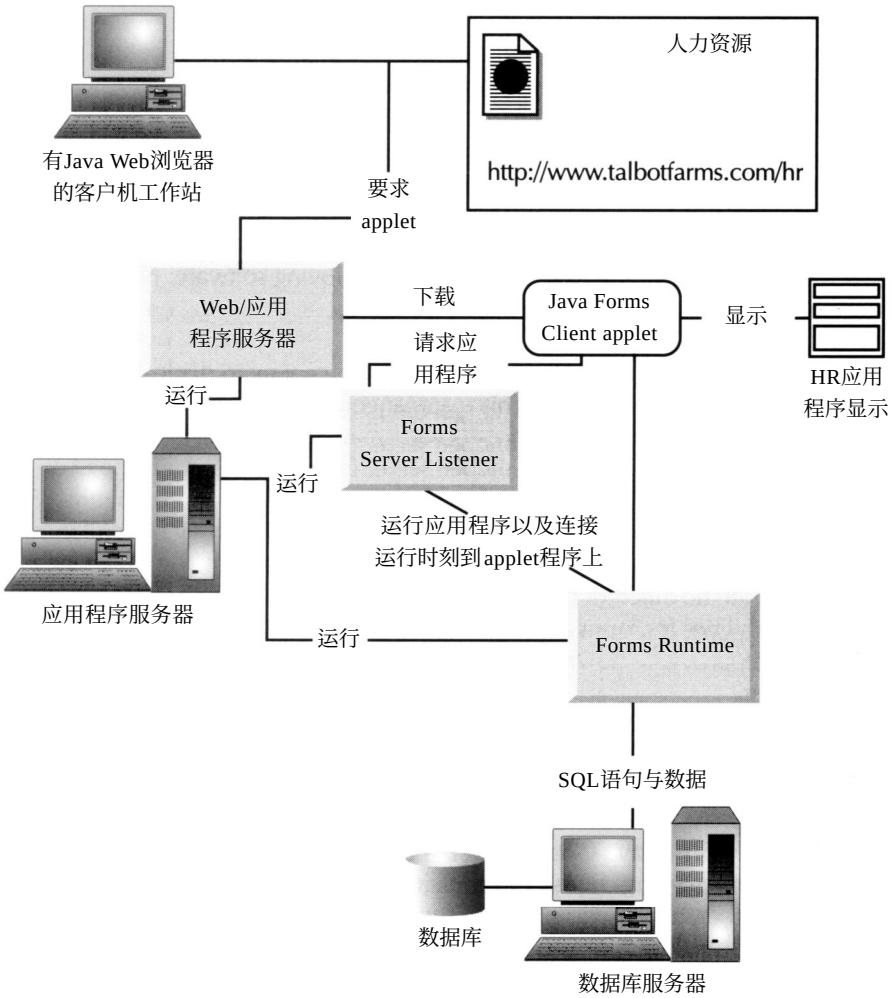


图14-1 Oracle Developer Server —— Forms Server的结构

这时Listener又将连接信息回传给小应用程序，与 Forms Runtime Server建立直接联系。小应用程序和Forms Runtime Server可以来回直接通讯，由小应用程序为表单提供显示功能，而

China-pub.com
下载

应用程序的其他事情由服务器负责。而 Listener则空闲下来接收其他应用程序发出的请求。Forms Runtime Server运行应用程序，通过SQL与Database Server进行通讯并获得数据，然后将数据传给小应用程序以供显示。

Reports则有一点不同。有两种方式进入 Reports Server：Reports Web Cartridge和Reports Web CGI。Reports Web Cartridge与Oracle Application Server相结合；CGI通过Web服务器与Report Server连接。图14-2说明了Report Server系统的结构。

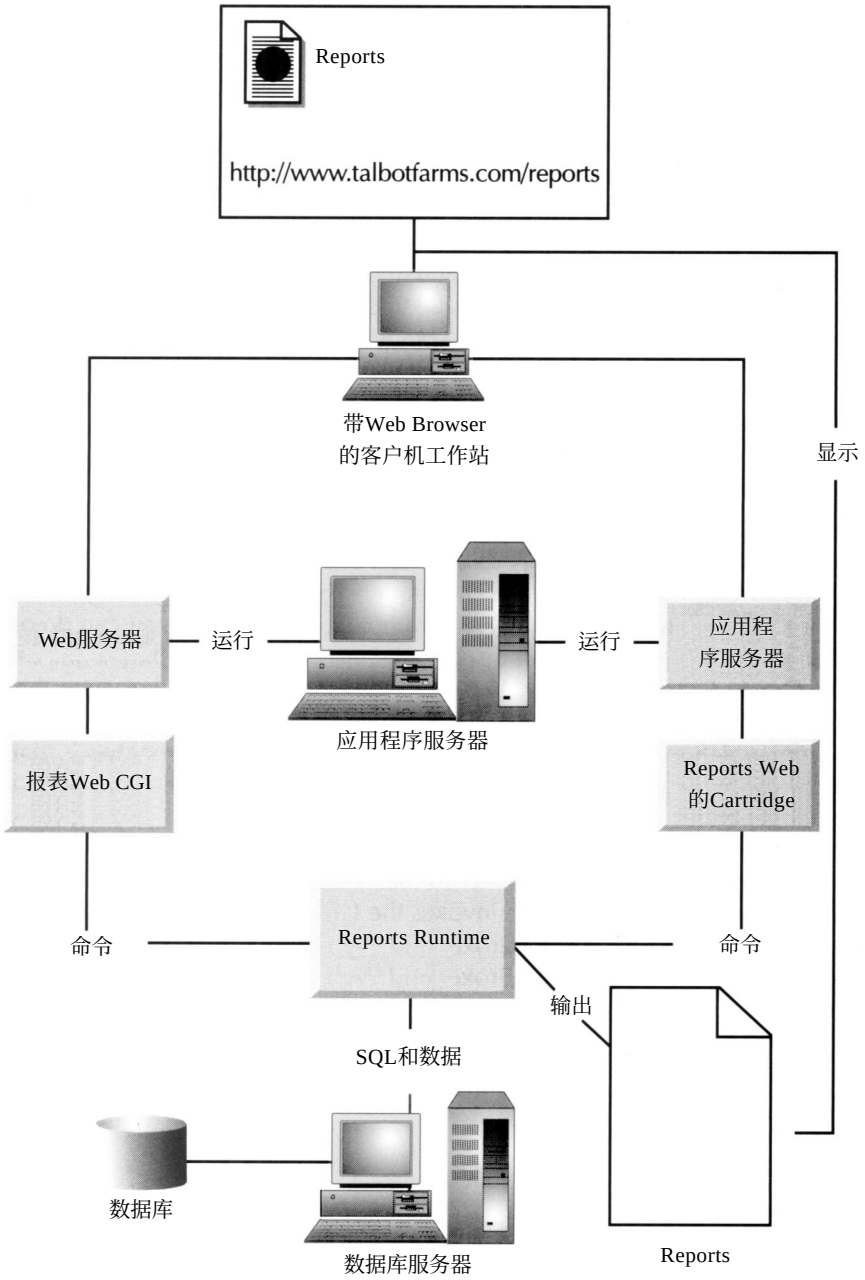


图14-2 Oracle Developer Server——Reports Server体系结构

China-pub.com 下载

客户机工作站运行Web浏览器，用户进入HTML网页可以获得报表(例如http://www.talbot-farms.com/reports)，轻点按钮或以超文本链接方式发出请求报表的信息。如果你用盒式磁带机方法进入，那么Application Server就会利用Report Cartridge建立一个Report Server命令行。Application Server接受命令，同时(等待输出)传给Report Server。Server又通过运行时系统运行报告产生输出文件，然后Web Cartridge将它传回URL告诉浏览器输出文件在哪里。或者你已经选用了CGI，那么Web服务器中的CGI脚本也可产生和提交命令行，然后告诉浏览器输出文件在哪里。浏览器可以HTML 、HTML CSS或Adobe Acrobat PDF方式显示报表。

Graphics的组织结构则更为简单一些，如图 14-3所示。

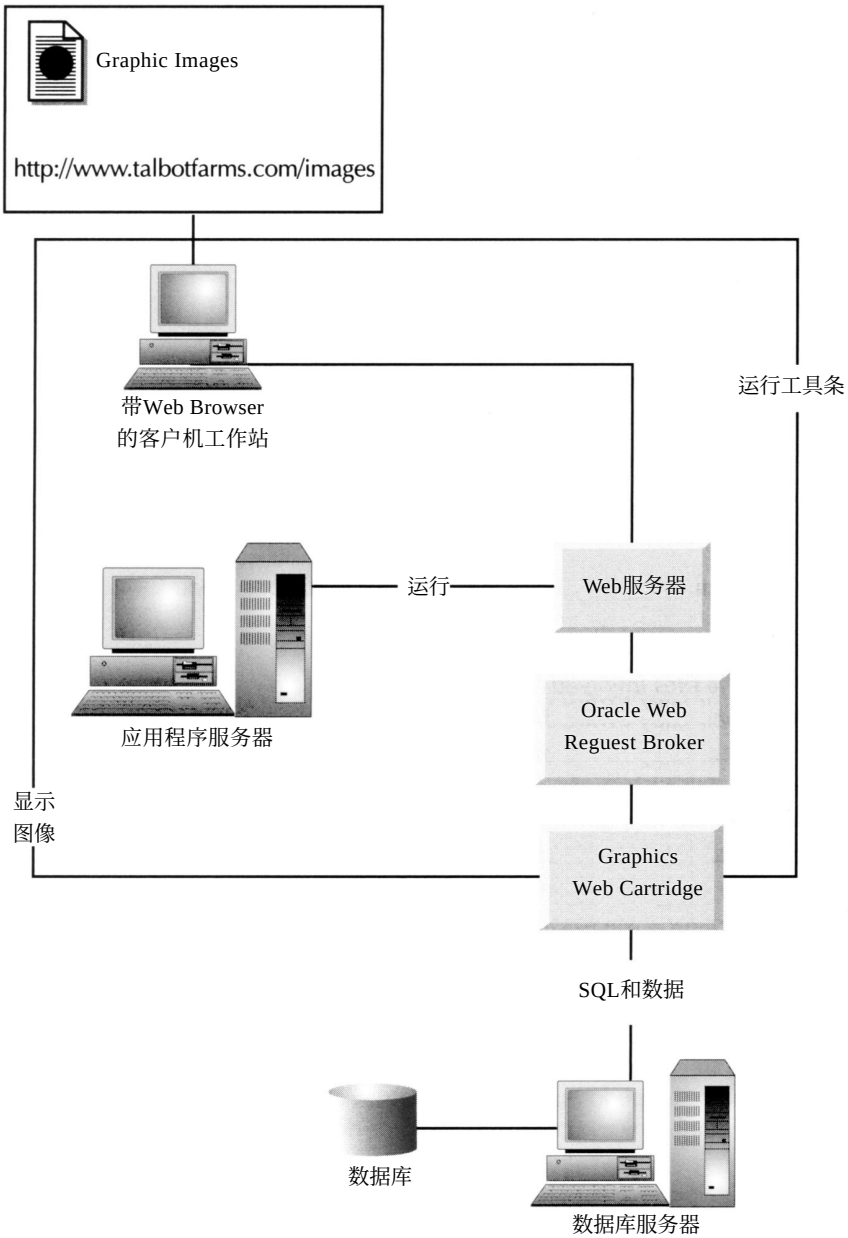


图14-3 Oracle Developer Server —— Graphics Server体系结构

China-pub.com
下载

客户机运行网络浏览器，用户进入他可以获得图形显示的 HTML网页(例如http://www.talbotfarms.com/images)。这时你可以任选一种方式：或是通过特殊图像显示链接，或是通过Oracle Graphics Builder Web界面的工具栏显示链接，工具栏为浏览器使用者提供了一个与数据库连接的方式，并显示一组可供选择的可用图像。这两种方式的区别在于利用本地 URL你必须自己设定URL的全部参数，而工具栏可为你自动产生全部参数，同时利用本地 URL在客户网页显示上你会受到更多的限制。不管你选用哪种方式， URL都要为显示图形向Web服务器发出请求，由Oracle Web Request Broker 或Graphics Sever Cartridge产生显示。Cartridge通过SQL进入数据库服务器为显示获取数据，然后Broker和Web服务器将图像传给网页。

14.2.1 安装Oracle Developer Server

安装Developer Server组件需要完成以下任务：

- 利用Oracle Installer提供的服务器部署选项，将Oracle Developer Sever从它的CD-ROM上安装到中间Application Server上。

注意 Oracle公司销售Oracle Developer Sever不同于Oracle Developer开发系统。

- 建立Forms Listener和Reports Server。
- 配置与Oracle Developer Server相连的Web服务器参数。
- 配置从服务器下载给客户机浏览器的HTML文件和其他环境选项。
- 用JInitiator软件配置客户机。

注意 涉及到不同服务器的多个文件放在Oracle安装目录层中，如果还不知道其中约定，那么Oracle Home就被认为是当前层的根目录，就像Oracle_Home一样，在后面所有的内容里会出现根目录。不论在哪里看 Oracle_Home，都会替代本地根目录，就像Microsoft Windows NT系统的G:\Orant一样。

在开始安装之前，必须先建立 Web服务系统。从理论上说，在多终端环境中使用任何网络服务系统都可运行Oracle Developer应用程序，但是采用Oracle Application Server可获得更好的集成环境和效能。每个 Web服务系统在建立虚拟目录时都有自己的原理，请参考服务器的指令文件。

注意 Oracle Application Server是一个在多终端环境中部署应用程序的常用应用程序服务器系统。请参考LLC的《Oracle Web Application Server by Dynamic Information Systems》(Oracle Press, 1998)一书。下面部分将重点放在Oracle Developer Server的Windows NT版和UNIX版的不同之处。

要在Web服务器上按照表14-1建立虚拟目录，在这里你可任取一个所喜欢的名字。

表14-1 为Oracle Developer Server设置的Web Server Virtual目录

/codebase/	物理层目录包含Java类文件，例如 Oracle_Home\forms60\java。如果你用了Forms Server中的平衡装载特性，那么在服务器池中的每个服务器中都要建立相同的目录。它允许盒式磁带机将服务器名称参数化，用最小装载服务器在运行时装入
/jars/	该物理层目录包含Oracle JAR(Java Archive)文件，通常与/codebase一样。在平衡装载服务器池的所有服务器上都要定义这个目录
/html/	该物理层目录用于Web服务器寻找静态的和盒式磁带机HTML文件

China-pub.com 下载

(续)

/cache/	该物理层目录包含Report Runtime的输出文件(仅用于Report Server)
/temp_image/	该物理层目录，如Oracle_Home\Forms60\Temp_Image，包含Forms Server向浏览器发送图像或位图时所产生的JPEG临时图像，在FORMS60_MAPPING环境变量(参见“配置环境”部分)中输入虚拟目录名，并在FORMS60_OUTPUT环境变量中输入物理层目录名
/web_reports/	该物理层目录包含Forms Server的输出报表，运行一个表单所产生的报表

1. 安装软件

在Internet上部署应用程序的第一步就是在中间层 Application Server上安装Oracle Developer Server。可以为你的Application Server选择一台机器，它只需要运行Developer Server即可。也可以将系统安装在常规的Web服务器上，但是会发现，加重常规Web服务器的负担会影响应用程序的效能，除非它可以装载大量的应用程序或你只有很少的用户。

如果有很多用户或很多应用程序，就会发现将进程分散到几个 Application Server上是很重要的。Oracle Developer Server包含一个平衡装载系统，它可以使你将 Forms应用程序进程装载到多台服务器上。

注意 有关建立平衡装载的信息请参考 Developer Server安装文件。有关利用Oracle Application Server初始化安装的信息请参考附录B。Graphics Server只在相关情况下工作，所以这一部分不讨论Graphics Server的安装。

1) 在你用来作为Application Server的机器上运行Oracle Developer Server安装文件，将会看到图14-4所示对话框。

2) 选择Deployment Installation单选按钮，会看到图14-5所示对话框。

3) 选择Server Install单选按钮，你就完成了安装，它将所有Oracle Developer Server的相关内容都安装到了Application Server上。

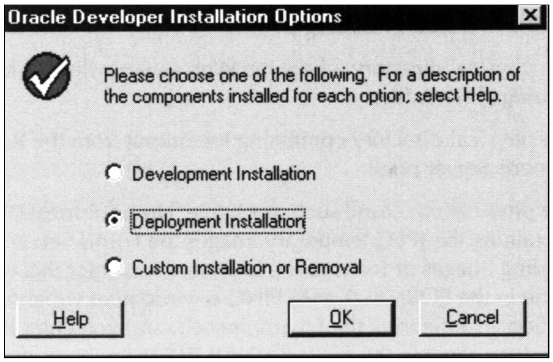


图14-4 安装文件对话框

2. 配置Forms Server

Forms Listener和Reports Server是两个必须手动开始的不同进程。当服务器开始后你可将Reports Server配置成自动开始。

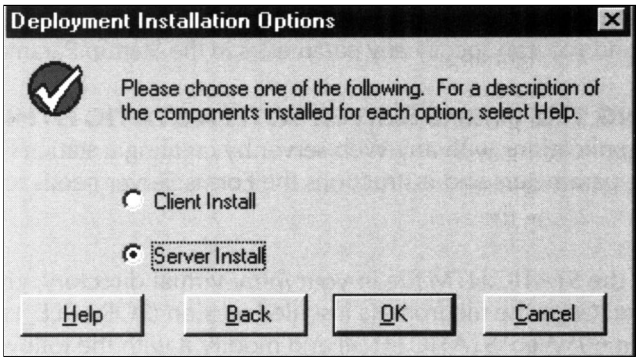


图14-5 Deployment Installation Options对话框

China-pub.com
下载

1) 从Windows NT任务栏启动Forms Listener：

Start|Programs|Oracle Developer 6.0|Forms Server Listener.(在UNIX环境下，发出命令ifsrv60-listen port=5555 &.)

2) 查看Task Manager Processes的内容,验证是否成功地启动了接收器 (在任务栏敲右键显示Task Manager)for IFSRV60.EXE。(在UNIX环境下，键入ps -ef | grep ifsrv60)。

3)在$ Oracle_Home\Net80\Admin目录下修改TNSNAMES.ORA，让它包含跟随TNS名称的规格，此时<hostname>为服务器名称或IP地址，如191.191.191.5：

```
repsvr.world=(ADDRESS=(PROTOCOL=tcp)(HOST=<hostname>)(PORT=1949))
```

4) 启动Reports Server作为NT服务(在UNIX背景使用命令“ rwmts60 name=repsvr.world & ”)处理，使用你添加到TNSNAMES.ORA中的TNS名字：

```
rwmts60 -install repsvr.world tcpip
```

5) 用Start|Settings|Control Panel|Services打开服务窗口，选择 Report Server服务，单击Start按钮。如果需要使用 Startup按钮使得服务器无论在哪里开始，都可自动启动服务，可以在Startup Parameters区设定任意参数。

3. 利用静态HTML页配置Forms Server

通过建立静态 HTML文件可在任意 Web服务器下运行表单应用程序，该文件包含 Forms Server需要的所有参数和说明。

- 在/html虚拟目录下建立STATIC.HTM文件，可任取名称。从安装版拷贝文件($ORACLE_HOME\Tools\Devdem60\Web\STATIC.HTM)，并用下面的内容替换修改：
- codebase：设置Web服务器时为Java编码定义的虚拟目录。
- archive：为建立应用程序下载的JAR文件的路径和文件，包括Oracle提供的ifsrv60.jar。
- serverPort：Forms Server Listener端口(默认值为9000)。
- serverArgs：表单参数和用户参数。
- serverApp：指定字体和位图路径设置的应用程序类的名称 (参看盒式磁带机文档)，“ default ”意味着使用默认的字体和图标路径。

下面是一个经修改过的运行LEDGER应用程序模板的例子：

```
<HTML>
<HEAD><TITLE>Talbot Farms Ledger</TITLE></HEAD>

<BODY><CENTER>
      <BR>Welcome to Talbot Farms.
      <BR>Loading application…
<P>

<APPLET CODEBASE="/codebase/"
      CODE="oracle.forms.engine.Main"
      ARCHIVE="/jars/f60all.jar"
      HEIGHT=20 WIDTH=20>
<PARAM NAME="serverPort" VALUE="9000">
<PARAM NAME="serverArgs" VALUE="Module=Ledger userid=Talbot/george@orcl">
<PARAM NAME="serverApp" VALUE="default"
</APPLET>
</BODY>
</HTML>
```

这一页将模块名称设置为 Ledger参数，所以运行附属程序 applet就运行LEDGER.FMX应用程序。需要为每个想运行的应用程序建立一个静态页。

China-pub.com 下载

你可能想尝试一种新的Oracle界面和感觉，为Oracle Developer Server运行应用程序提供不同的界面，那么为HTML添加以下代码：

```
<PARAM NAME="lookAndFeel" VALUE="oracle">
<PARAM NAME="colorScheme" VALUE="teal">
```

可以使用Oracle指定的Windows组件中不同的颜色对colorScheme值来改变颜色。

(1) 用JInitiator配置客户机

Oracle JInitiator是一个Oracle 公司的Java插件，用于在HTML页中运行Java小应用程序或JavaBeans。它为优化使用Oracle Developer应用程序建立一个特定的Jave 虚拟机器。可以让用户在运行时从他们的客户机上下载，也可以预先在客户机上安装。

在Oracle Developer安装处(Oracle_Home)找到jinit.exe文件，将它拷贝到客户机上，并运行该程序。这是一个可执行的用于安装Java插件的安装保护程序。

(2) 在因特网上使用图标和字体的应用程序类

在因特网上部署表单，在网络上使用的GIF图像和字体是一个难题。如果想定义特殊的字体和图标，可使用标准的Form Builder来设定。但是如果在一个题目中有若干次变化，就可能需要使用应用程序类。这种情况多发生在不同 GUI平台上(如Windows、Motif和Macintosh)需要特殊字体的情况中。

一个应用程序类就是特殊字体的一组字体映像的集合和指定图标的图标路径。可以在几个依赖于 HTML serverAPP 参数的配置文件或静态HTML基本文件中之间转换(有关参数的详细内容见“利用静态HTML页配置Forms Server”一节)。运行时，Forms Server首先使用已命名的应用程序类的设置，如果在 serverApp参数中没有命名应用程序类，那么 Forms Server就使用默认的字体和图标。默认字体建立到表单组件的环境中，默认图标从与基本 HTML页相同的目录中获取。

建立应用程序类需要编辑REGISTRY.DAT文件，可以从以下目录中找到：

```
Oracle_Home\forms60\java\oracle\forms\uiClient\v1_4\util
```

1) 对想添加的每个字体的映像图，在 REGISTRY.DAT文件中加一行，包括应用程序类名称、字体参数名称、字体参数值，如下面的 Windows_Form应用程序类：

```
Windows_Form.appFontNames=Courier,Terminal,Arial
Windows_Form.javaFontNames=MonoSpaced,Dialog,Helvetica
```

2) 当没有字体与应用程序字体相配时，为了使表单显示指定的默认 Java字体，应在Std_Form应用程序类增加以下几行：

```
Windows_Form.DefaultFontname=Helvetica
Windows_Form.DefaultSize=12
Windows_Form.DefaultStyle=BOLD
```

3) 要指定一个图标目录，应在Windows_Form应用程序类中增加以下一行：

```
Windows_Form.iconpath=/temp_image
```

在这种情况下，该值是一个虚拟目录 /temp_image/，你已经将它定义在了 Web服务器上，参看表14-1。也可以指定一个URL或一个与HTML页相关的目录，而虚拟目录是最佳的选择，因为在移动它而没有改变REGISTRY.DAT文件时，可以在Web服务器上改变它。

(3) 配置Reports Server CGI

• 配置Reports Server CGI是简单的：

China-pub.com
下载

在Oracle_Home\bin下找到rwcgi60.exe文件，将它拷贝到Web服务器CGI脚本目录中，如Oracle_Home/ows/40/bin(如是UNIX环境，拷贝rwcgim60文件)。

4. 建立Reports Server和Forms Server作为NT服务

现在已经安装和配置了服务器，但是还需要将它们设置 NT服务(或UNIX后台进程)。这里最好能够用管理员身份注册到NT系统上，参阅NT系统管理员帮助。

1) 首先为Reports Server建立一个TNS项。编辑TNSNAMES.ORA文件，如果是Oracle 7 SQL*Net，则该文件在Oracle_Home\Network\Admin目录下；如是Oracle 8 Net8，则该文件在Oracle_Home\Net8\Admin目录下，将TNS项添加在文件底部：

```
repserver.world=(ADDRESS=(PROTOCOL=tcp)(HOST=<server>)(PORT=1949))
```

这里，<server>是运行Reports Server的服务器名称或 TCP/IP地址，如 WAVERLY或127.100.100.100。只要你添加了适当的区域，就可以使用任何 TNS名称，如reports.world或news_of_the.world。

2) 为Forms Server创建NT服务。应当以管理员身份注册进入 Windows NT。在任务栏中以Start|Programs|MSDOS Promt打开DOS shell，执行下面命令：

```
ifsrv60 -install forms60
```

3) 在相同的DOS shell下为Reports Server创建NT服务，命令如下：

```
rwmts60 -install repserver.world tcpip
```

服务器名称repserver.world，是你在TNSNAMES.ORA文件中指定的TNS名称。

4) 运行Service(服务)小应用程序：选择 Start|Settings|Control Panel，然后双击Services图标，找到Forms Server 60入口，并点击Startup。在得到的对话框中(见图14-6)，找到Log On As部分，选择This Account，填写当前用户名和口令，即在当前帐号下运行服务。也可选择 System account为系统帐号(这是个好选择)或一些其他的帐号，可参考管理程序。最后单击 Automatic，让NT在引导时自动启动 Forms Server。

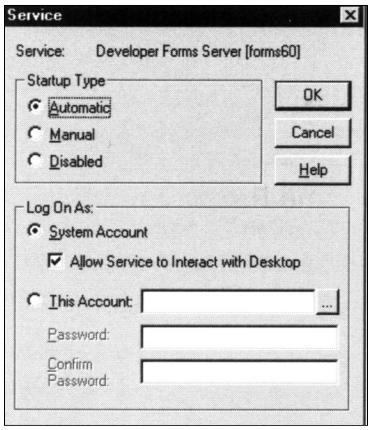


图14-6 Service对话框

提示 在系统帐号环境下和其他标准服务一样运行这种服务，通常是个好方法。可以简化安全性和配置问题。然而，也可能会因为安全或其他原因不能为特定的NT服务器使用这种方法。

5) 在Service对话框中重复最后一步，建立Report Server60登录项。

注意 你也许需要阅读有关创建NT服务的帮助文件，以便解决在建立Reports Server时遇到的更广泛的问题。因为有更多的需要考虑的问题超出了基本安装范围。

5. 配置环境

一旦建立了服务器，就必须为它们配置各种环境选项。在 Window NT环境下就意味着编辑系统注册表。

1) 在任务栏中选择Start|Run，输入下面命令并单击OK：

```
regedit
```

China-pub.com 下载

2) 找到注册表树下的 HKEY_LOCAL_MACHINE|SOFTWARE|ORACLE目录，单击ORACLE，显示所有Oracle关键字(见图14-7)。然后建立表14-2中列出的关键字。

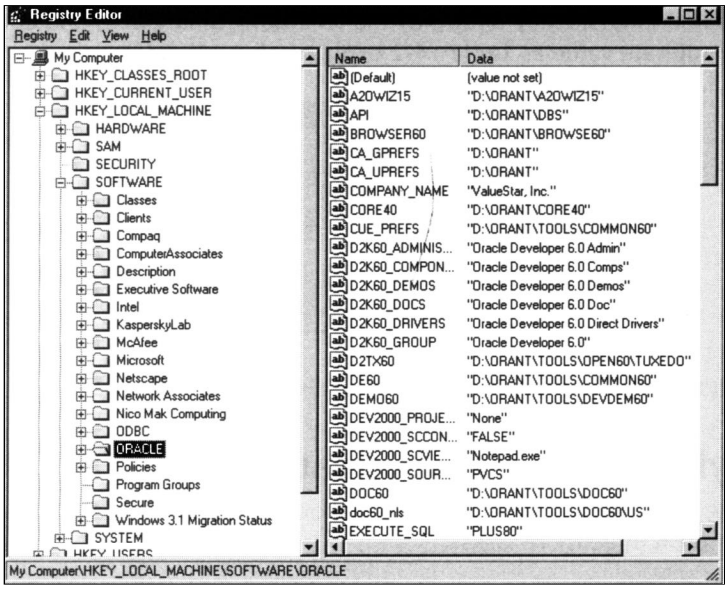


图14-7 显示Oracle关键字

表14-2 Oracle Developer Server的注册关键字

名　称	类　型	值
FORMS60_MAPPING	字符串值	如果在应用程序中使用了图像和图标，那么必须在Web服务器上建立临时目录和虚拟目录，当把它们传送给客户浏览器时，Forms Server这些目录来临时存储JPEG图像。键入目录名称，如“/image,”(结尾处无斜线)
FORMS60_OUTPUT	字符串值	一个对应FORMS60_MAPPING虚拟目录的物理目录，如G:\Orant\Forms60\temp_image
FORMS60_PATH	字符串值	在服务器上以分号隔开的物理目录列表，Forms Server为可执行的FMX表单访问它们。Oracle安装程序应当在初始化时自动建立这个列表，所以在这里只需要进入指定目录
FORMS60_REPFORMAT	字符串值	使用字符串“HTML”告诉Forms Server产生类似于HTML页的报表，用“PDF”则产生Adobe Acrobat报表
FORMS60_TIMEOUT	双字值	键入Forms Server等待输入所允许的最长分钟数，应为3–1440之间的整数，可从15分钟开始
REPORTS60_WebLOC	字符串值	Web服务器查找报表输出文件的虚拟目录，如/web_reports/
REPORTS60_WebLOC_TRANSLATED	字符串值	如果Reports Server和Web服务器不能共享一个文件系统时，Reports Server用来转移Reports Runtime输出的物理目录，如G:\ORANT\web_reports
REPORTS60_PATH	字符串值	Reports Server查找报表定义文件的物理目录列表

China-pub.com
下载

对表中的每个关键字，都要执行以下步骤：

3) 右击关键字，显示面板，选择 New|String Value或New|DWORD Value，如表14-1所示。Editor创建有关重新命名的新关键字和位置(见图14-8)。

图14-8　新关键字及其位置

4) 按照表14-2所示的关键字重新命名。

5) 双击关键字，显示 Edit String对话框，按照表 14-2的说明填写适当的值，然后点击OK(见图14-9)。

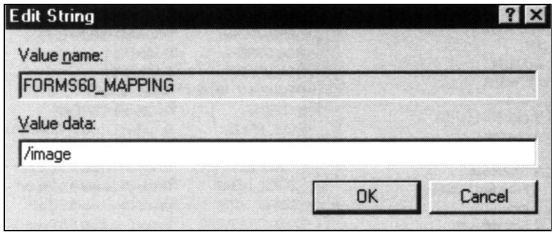


图14-9　Edit String对话框

6) 用File|Exit退出注册编辑器Registry Editor。

14.2.2　部署因特网应用程序

要完成使用Web浏览器部署应用程序，还必须完成下面三个任务：

- 在服务器上安装表单、报表和显示的应用程序代码。
- 在服务器上安装HTML访问页。
- 为用户提供访问页的URLs。

编译应用程序的输出由可执行文件组成（FMX、RDF和OGD文件)。必须用与Developer Server所使用的相同的操作系统的生成器或编译器来编辑这些可执行文件。例如，如果准备在Windows NT环境下运行Developer Server，那么必须用Windows NT的生成器或编译器来编辑应用文件，而不能用Windows 98或Sun Solaris。

要安装表单的代码 (FMX可执行文件)，将它们放入所希望的任意目录下，在动态或静态的HTML页上用完整的路径和文件名使用它们。要安装报表代码 (RDF可执行文件)，也可以将它们放入服务器的任意目录下，但是它在服务器注册表中的变量应为 REPORTS60_PATH。

注意　可执行的Graphics必须放在可知位置，允许Graphics Server以图形显示方式的下拉列表提供任务栏。

将动态或静态的 HTML基本页放入标准虚拟目录中，如 /html，然后为用户提供适当的URL。例如Talbot 农场有一个WorkerSkill表单静态基本页，叫WorkerSkill.html，Talbot用户在URL上写下面命令即可运行表单：

```
http://www/talbotfarms.com/html/WorkerSkill.html
```

14.3　客户机/服务器计算领域

在多层服务器上部署应用程序具有巨大的优势。然而，如果因为某些原因，环境要求单一服务器，那么还可以像客户机 /服务器应用程序一样部署应用程序，在每个想要运行应用程

China-pub.com 下载

序的客户机上安装Oracle Developer Deployment软件。本部分将介绍建立客户机/服务器软件包的步骤。

用户必须从数据库或系统管理员处获得一些专门的信息，比如主机和网络文件服务器的连接字串和数据库服务器名称。必须为用户提供文件和使用材料，必须告诉用户什么信息是他们必需的，还需要仔细告诉用户在所指定的环境下所要求的附加软件（包括Oracle Developer运行时和Oraclr SQL*Net软件），以及所需要的特定版本。

14.3.1 将应用程序打包

在建立应用程序安装系统时，有两方面的情况决定你的努力成果：安装过程中个人参与的程度和安装过程的复杂性。

通常，个人参与程度和应用程序本身的性质直接相关。你自己安装的应用程序通常是个人性的，或是某个单位的工具程序，或是某个工程的应用程序，而公司社团的应用程序倾向于要求一些个人参与在其他的应用上使用的功能。

安装过程的复杂性包含几个方面。第一，可能需要将应用程序分成几个部分：客户工作站和服务器、一些协作工作站、一些分散的服务器、一个 Web服务器等等。第二，可能需要有条件地安装程序包的各个部分。例如用户可以选择只安装程序包的一部分，或者在安装过程中检测到的指定配置的那一部分，或者检测到的需要更新的那些部分，或者需要重新安装的部分。第三，可能需要对程序包进行一些处理：解压缩；检查哪些程序是已存在的，哪些是新的；编译；生成选择文件；或其他一些对用户来说增加了复杂性的安装方法。

如果产品在安装过程中没有个人参与并有相当程度的复杂性，那么应该使用专业的安装软件程序包。这些程序包提供了所需要的全部工具，对最终用户和其他管理者即购买产品的人来说，再复杂的安装也会显得简单和专业化。

否则，你必须留心每一个细节，必须了解每个地方所需要做的和已经做的所有事情。程序打包的总原则就是世界上最古老的原则：力求简单，仅安装用户需要的文件。不要用一些东西将磁盘塞得乱七八糟，如压缩说明文件，产品不同部分的 23个README文件，用户可自己定制安装的600个特殊图标位图。将所有这些放在以备特殊需要时使用的磁盘上。

此外，必须决定是否将源文件和目标文件一起提供，这个决定通常取决于你和用户之间的合作性质。如果需要让用户防止他们的错误，不要提供源程序。如果想要保护你对该系统的专有权，不要提供源程序。如果用户不需要或不想要源程序，不要提供源程序。

这里是Oracle Developer应用程序打包组建清单，表 14-3告诉你哪些组件是必须打包的，哪些组件是可以选择的。应该仔细地将这些文件的用途和特性写在 README文档中，并将它装入软件包中去。此外还可以将各种各样的文件与系统文件一起打包。例如图标、位图、声音文件及视频文件等等。

表14-3 Oracle Developer文件组件

产品	文件	源程序或可执行程序	可选择的或需要的	说明
应用程序系统	PRF	源文件	可选择的	如果这些与Oracle用ORAINST建立的默认值不同，那么用应用程序建立一个模块优先文件或建立一个系统运行所需要的包含运行时间选项的文件

China-pub.com
下载

(续)

产 品	文 件	源程序或可执行程序	可选择的或需要的	说 明
	EXE	可执行文件	需要的	被允许的Oracle运行时间可执行文件，一些增加的可执行工具。用户可使用ORAINST安装，它们在Oracle公司提供的安装光盘上
	DLL	可执行文件	需要的	通过ORAINST安装的被允许的Oracle运行时间动态连接库，和其他一些应用程序所需要的库 (如用户退出、外语功能等)
表单	FMB	源文件	可选择	表单源文件。只在想要用户能够修改应用程序时才装载
	FMX	可执行文件	需要的	没有源程序的可执行代码
菜单	MMB	源文件	可选择的	菜单文件源代码。只在想要用户可以改变应用程序菜单时装载
	MMX	可执行文件	需要的	没有源程序的可执行菜单代码
PL/SQL	PLL	源文件	可选择的	PL/SQL库源程序。只在允许用户改变程序单元时装载
	PLX	可执行文件	需要的	库中的可执行程序单元
报表	RDF	源文件	可选择的	报表源文件。只在允许用户改变报表时装载
	REP	可执行文件	需要的	可执行报表
图形	OGD	源文件	可选择的	图形显示源文件。只在允许用户改变显示时装载
	OGR	可执行文件	需要的	可执行显示文件

注意 可执行文件必须和用户的操作系统兼容，这意味着必须在该操作系统下编译源文件。例如不能在UNIX系统中编译一个表单，又将生成的可执行文件交给Windows 98用户去执行。

还需要考虑需要提供哪些文件用于升级更新。你可以提供一个完整的系统，包括所有的文件，也可以只装载改动了的新文件。如果选择后一种方法，那么将需要扩大系统测试涵盖所有可能的不同配置，还必须人工组成 Oracle Developer Server的Project Builder的内容，仅提供你所指定的文件。必须保证文件的子集对系统的所有不同配置都能够服务。

注意 作为一个实用而合法的软件，必须用Oracle Installer来安装客户机/服务器的配置选项。有关安装配置软件的详细内容请参考附录B。

可以点击图标并输入适当的信息，使用 Form Compiler(表单编译器)产生FMX文件。然而还有更简单的方法，通过 Project Builder，只需单击一下鼠标就可容易地编译所有文件。你需要按照第3章中“ Oracle Developer工程管理器 ”部分讨论的那样建立工程文件。首先在 Project Navigator中找到想要编译的目标代码节点，选择该节点，然后从这个结点的弹出菜单 (点鼠标右键)中选择Build All菜单项，或在主菜单中选择 Tools |Build All菜单项(见图14-10)。

可以只产生用于文件和压缩的 FMB和MMB文件的文本。 Oracle Developer支持扩展编辑以缩短生成所需要的时间。如果你对一些 PL/SQL做了小小的改动，然后重新生成表单，那么Oracle Developer不会重新编译表单中的所有 PL/SQL，而只重新编译改动的触发器或程序单元。为此， Oracle Developer将附加信息存在 FMB文件中，如所有的 PL/SQL源文本，中间描述代

China-pub.com 下载

码和已编译的代码。只有在生成表单或编译触发器或程序单元时，才生成中间描述代码和已编译的代码。变换成 FMT文件然后又变换回 FMB文件回缩小 FMB 文件。因为FMT文件不存储中间描述代码和已编译的代码，所以，生成的 FMX文件仅包含已编译的代码，这就是要使用的文件。

在Form Builder中，可以使用File| Administration|Covert菜单项生成FMT、MMT表单和菜单的源文件文本。这个菜单项显示在Covert对话框中(见图14-11)。

要生成FMT表单源文件，将 Type设为Form，浏览表单文件(FMB)，并将Direction域设置为Binary-to-Text。要生成MMT菜单源文件，将Type设为Menu，浏览菜单文件(MMB)，并将Direction域的内容设为Binary-to-Text。当单击Convert时，Builder就会生成表单源文件和菜单源文件。

注意 可以用Form Compiler生成表单和菜单的文本文件。选择“ Write script file? ”复选框，将告诉你生成的是文本文件而不是目标代码。

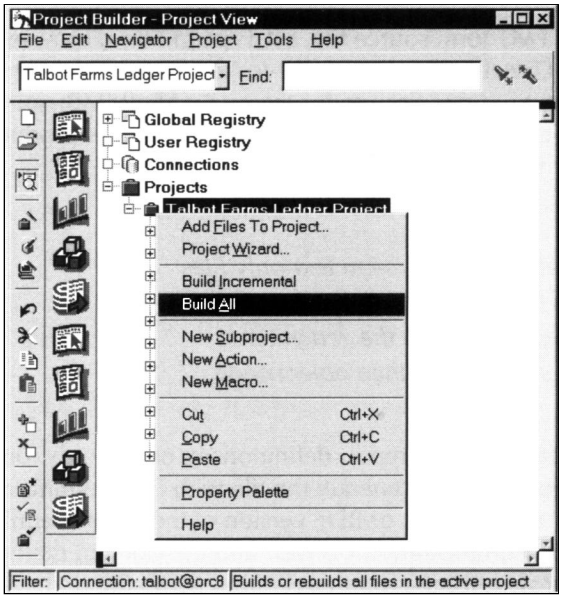


图14-10 使用Project Builder编译文件

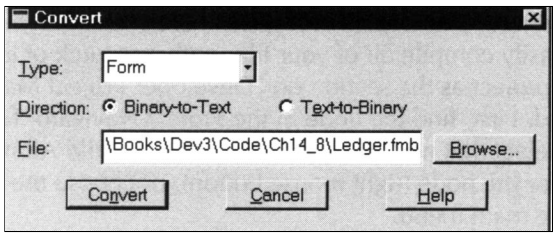


图14-11 Covert对话框

在Report Builder中有两种方式产生REP文件(报表定义为单一目标格式)。可用File|Administration|Generate生成这个文件，也可以通过File|Administration|Convert和Convert对话框将数据库或报表定义的 RDF版本转化为REP版本。将Source Type设为RDF，在源文件域浏览报表RDF文件，将Destination Type设为REP，并为Destination域填写文件名。可以用Convert对话框创建生成REX文件(报表定义为单一源文本格式)，通过选择REX的Destination Type取代REP(见图14-12)。

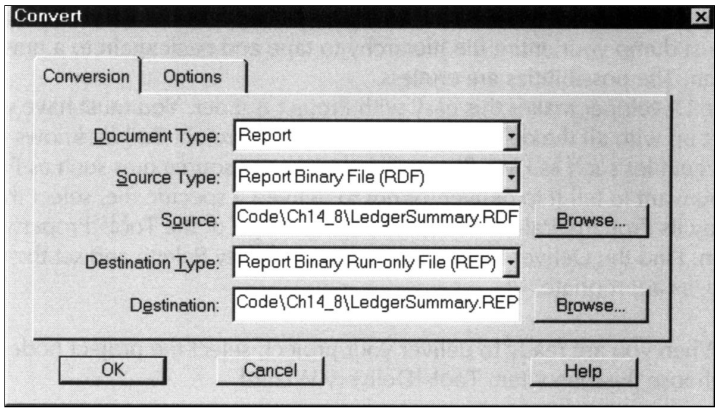


图14-12 选择Destination Type选项

China-pub.com
下载

可用File|Administration|Generate菜单项生成OGR文件(单一目标代码图形显示)。

14.3.2 客户机/服务器的部署

可以用很多不同的方法为客户部署你的程序包。但在不久以前，还只能根据目标操作系统或其他系统选择软盘或磁带。

目前，CD-ROM得到了偏爱，特别是对 Oracle或Oracle Developer这样大型的商业产品。CD-ROM作为主要工具，随着其价格的很快下降和技术上的进步，你应该认真考虑这种选择。

不久以前，因特网和其他在线服务和商用公告栏已经成为获取软件包的资源。可以通过连接到适当的服务器和FPT或将程序包下载到工作站上，然后安装。这是发送低价软件、免费试用软件和升级的普遍方法。如果你的公司全部连接到了因特网或本地网或广域网络，那么可以用这种机制有效地发送家用应用软件。

更新的Web 技术使你下载程序更加容易并可运行，如 Java，在前面部分有详细介绍。

大多数商业安装产品是通过这些不同的媒介发送的。例如用软盘，在盘上系统提供压缩和打包文件，并把建立和压缩文件的过程作为在工作站上安装的一部分。可以用标准的磁盘压缩工具简单而廉价地打包文件，如 PKWARE公司提供的各种各样的 PKZIP压缩工具。若使用类似于磁带机的大容量媒介，则只需将大量文件分层次地装入磁带机并重新返回到一个新的工作站上。这种能力是无止境的。

Oracle Developer利用Project Builder将工作简单化。必须用所有能够识别的可发送文件建立工程。Project Builder知道传送类似于FMX文件的目标文件，而不发送像FMB文件这样的源文件。如果想要发送或不发送一个指定文件，可以利用弹出菜单或用 Tools|Property Palette菜单选项，选定该文件并显示它的 Property Palette。Property Palette中找到Deliver文件属性，安装需要设置值为Yes或No。

1) 当准备发表工程时，选择这个工程的结点并选择菜单项Tools|Delivery Wizard(见图14-13)。

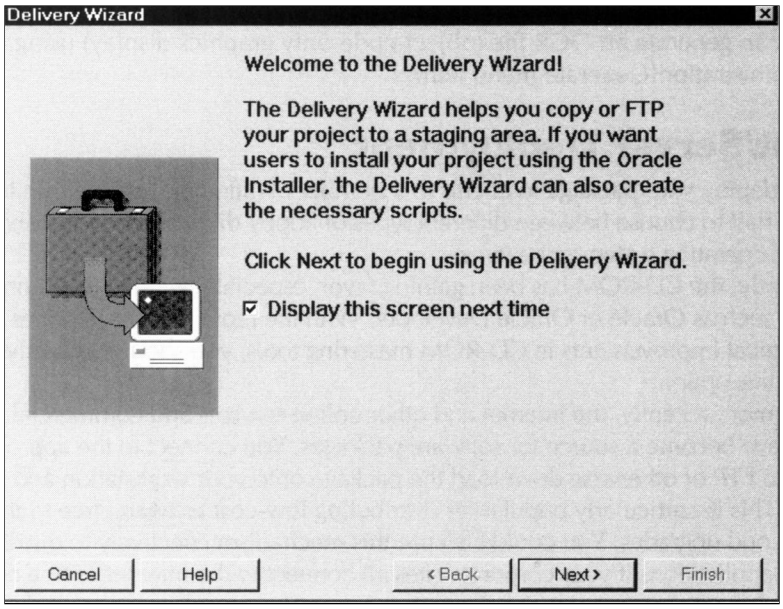


图14-13 Delivery Wizard对话框

China-pub.com 下载

这个Wizard将工程中可以发表的部分拷贝或FTP到公布区域。还可以让向导创建脚本，这样可以通过Oracle Installer将工程文件和运行时间安装结合起来，这使你非常容易地部署Oracle Developer运行时和应用程序代码。

使用Delivery Wizard有两种方式：发送一个完整系统或发送一个更新系统。下一屏让你选择是发送目标文件中所有可发送文件还是只发送至上次发送后改变的文件。

2) 单击Next。如果要发送一个完整系统，那么选择“Deliver all files”，否则选择“Deliver files that have changed since the last delivery。”(见图14-14)。

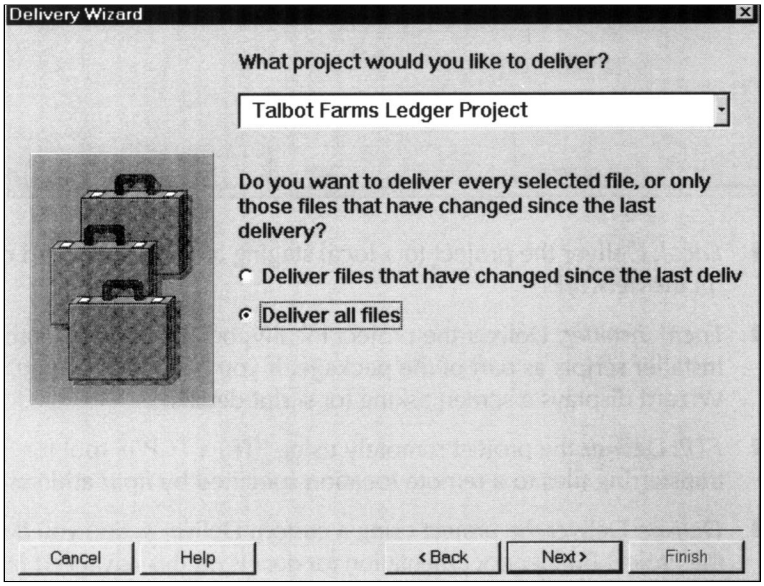


图14-14 选择Deliver all files选项

3) 单击Next。选择想要建立的发送类型，然后输入分级区域的路径或FTP地址(见图14-15)。

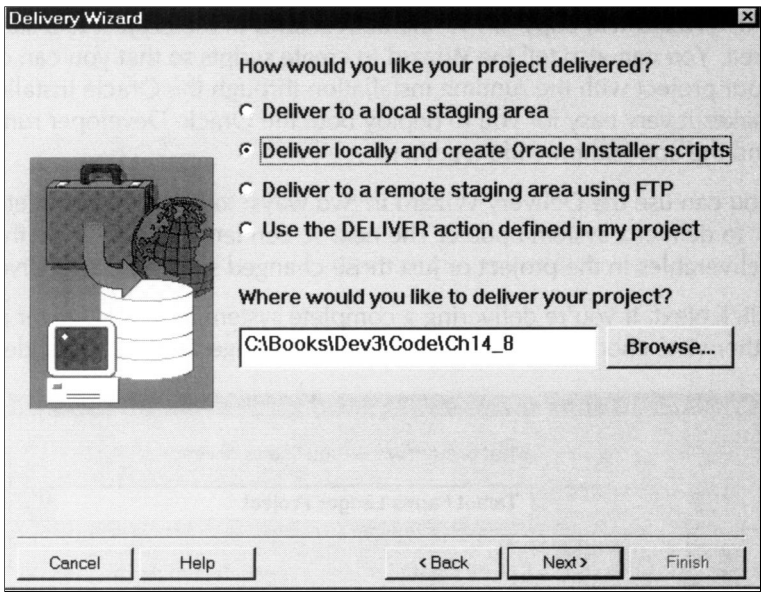


图14-15 输入分级区域的路径或FTP地址

China-pub.com
下载

■ 本地的(Local)：将工程文件发送到你的本地硬盘或网络的合适区域。

■ 本地安装程序(Local installer)：本地发送工程文件并创建Oracle Installer脚本作为程序包的一部分。如果你选择了这一选项，那么Wizard就会显示下一步要求更多的细节。

■ FTP：应用FTP远程发送工程程序，一个TCP/IP传送工具远程发送文件到由ftp://address指定的位置。

■ Deliver：使用定义的Deliver方式发送工程程序(参见Project Builder文件)。

4) 单击Next。在文件屏幕所列出的文件中选出将要发送的工程文件。通常你将会发送所有文件，那么单击Next即可(见图14-16)。

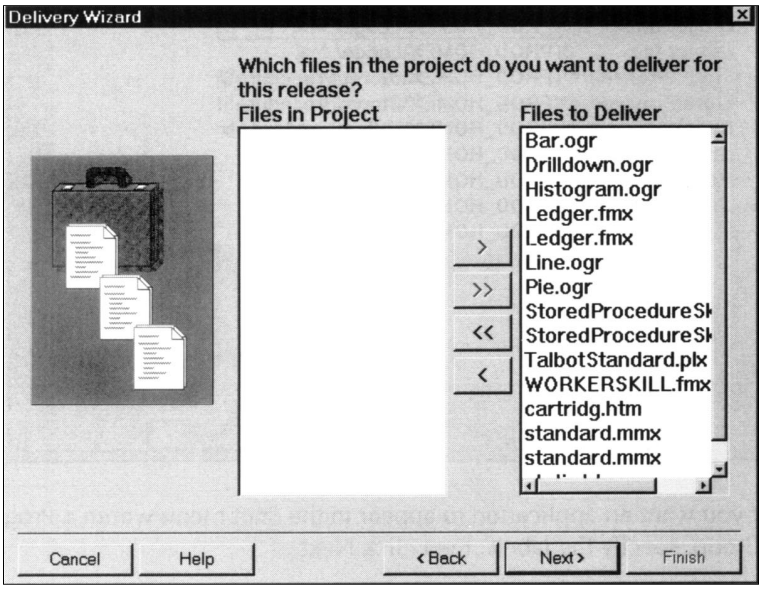


图14-16　选择Next，发送所有文件

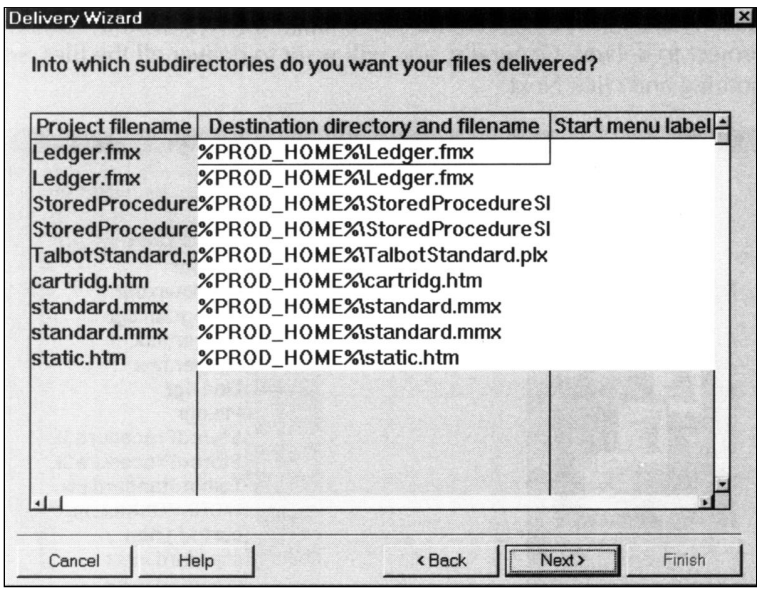


图14-17　第一个屏幕窗口

China-pub.com 下载

现在Wizard进入产生Oracle Installer脚本过程。第一个屏幕窗口显示想要安装文件的位置。通常应该将所有的工程文件安装在一个单独的目录下。不这样做，就意味着大量的文件将会随处放置(见图14-17)。

5) 如果想要应用程序出现在 Program Group的Start菜单中，那么指定标签，然后单击Next。

6) 单击Finish，发送工程文件(见图14-18)。

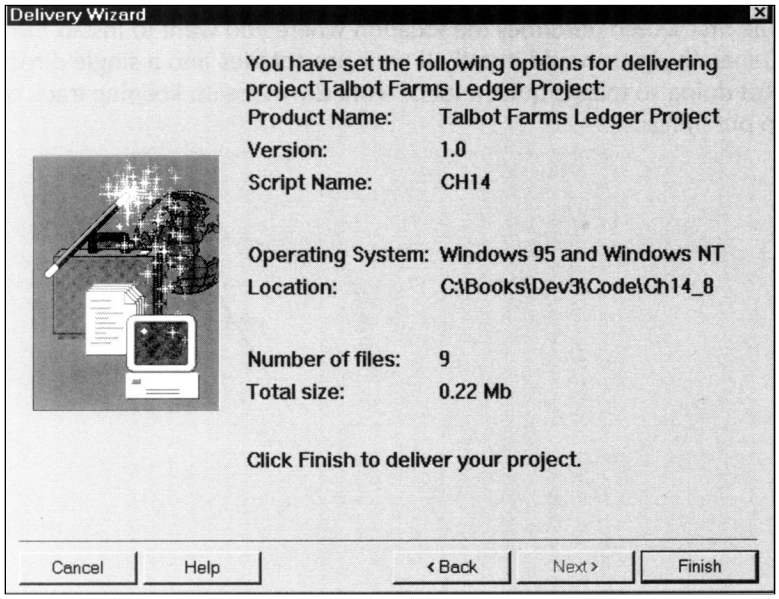


图14-18 选择Finish发送工程文件

现在已经完成了发送工程文件的工作。下一步是将文件压缩为一个 ZIP文件或其他形式发送到软盘、网络或CD-ROM上。下面讲述发送部署以后如何进一步管理。

14.4 配置管理

当进入了应用程序的生命周期阶段时，使应用程序在配置管理系统的控制下是非常重要的。如果脱离了配置管理系统的管理或控制，那么就会受到损害。在部署过程的每一步都应当重视这种形式的控制，包括控制所提出的要求和设计方案。

Oracle Developer结合了多种版本控制和配置管理工具，如 Intersolv PVCS、Rational Clear Case或Starbase中的Starteam。你可从Oracle Developer Server的每个单一组件的生成器中使用这些工具，但是你必须用 Project Builder界面来管理文件配置。可以使用 File|Administration菜单的Check-In、Check-Out，Source Control Options选项或这些选项的弹出菜单中的 Check In和Check Out选项。

界面产生 ASCII文件，存储在版本控制档案中。可以通过 File|Administration|Source Control Options菜单选项和Source Control Option对话框创建档案，在对话框中指定档案目录的位置和名字。

File|Administration|Check-In菜单选项显示Check-In对话框，它让你选择一个模块放入档案。File|Administration|Check-Out菜单选项显示Check-Out对话框，它让你选择一个模块从档

China-pub.com
下载

案中取出。如果想修改模块，必须选择 Lock This File复选框，否则得到的是只读模块文件。

Oracle Developer的配置管理生命周期有以下几个过程：

1) 创建档案目录。

2) 选择模块的初始版本。

3) 选择想要修改的模块。

4) 选择已经修改过要返回归档的模块，这些模块使用新的版本号。

使用配置管理软件窗口，可以管理配置管理系统中更高级的特性，如跨越几个模块发表版本基准。也可以用 File|Administration|Source Control Options菜单选项和 Source Control Options对话框产生不同的报表(报表显示两个文件的不同)。

也可通过输出文本文件来利用其他配置管理系统，如 Source Integrity on Windows或UNIX中的rcs和sccs，虽然不能直接通过 Oracle Developer记录和检查文件。如果系统提供了类似于PVCS的命令行，就可以在 Project Builder中增加一些基本特性。这个小小的缺点不会有太大的影响。

在大多数配置管理系统中版本号相当复杂。对大多数应用软件来说，一个简单的两部分组成的版本号是有效的： <Release>.<Version>。<Release>号随产品主要更新而变化，<Version>号随每个升级而变化。例如1.0是产品最初的版本，1.5是第5个较小的升级，而2.3是Release 2主要版本后的第3个较小版本。“可能是(Might be)”描述了所有的情况，因为你不可能依靠版本号这种简单的逻辑。这里涉及到管理。你将发现跳过较小版本的很多理由，在两个主要版本号下有并行版本，还有许多其他混乱但是合理的情况。 3000年后这将为上百个考古学家和历史学家提供工作机会，因为他们要找出计数系统的原始软件。

现在你已经学习了使用 Oracle Developer的部署和使用的所有基本组件，最后一章将学习应用程序与Oracle Developer产品之外的工具相结合的先进工具。

China-pub.com
下载

第15章　Oracle Developer与开放式系统

开放式系统是指允许使用来自不同的制造商的工具的系统。 UNIX成为了原型的开放式系统：有许多不同的UNIX操作系统，可以互换地使用它们 (当然要合理地使用)。在20世纪80年代后期，随着库技术和工具集成体系结构的发展，工具集成开始在软件工具市场出现，例如Atherton Backplane，它允许程序员把多种软件工具集成到一个聚合的工作过程中。后来针对PC机和Macintosh机又出现了动态数据交换 (DDE)、发布(Publish)和预定(Subscribe)，允许系统设计人员在运行时与其他的工具进行通讯。

一方面，随着对象链接和嵌入 (OLE)接口技术的发展以及 ActiveX和组件对象模型(COM)的成功实现，另一方面，随着 CORBA(公共对象请求代理体系结构)和网络化的Java(“可在任何地方运行”)的出现，开放式系统开始向建立软件体系结构方面发展，即向下至对象层开放所有方法。使用这样的体系结构，可以创建对象并在许多应用程序和服务器中共享使用它们。

本章介绍Oracle Developer提供的开放式系统环境中的一些方法。现在达到使应用程序脱离真实世界而只作为一个系统的阶段。如何能够使用所有的 Oracle Developer共同工作来开发应用程序呢？对Oracle Developer以外的对所有系统要做些什么呢？如何能创建一些东西作为一个系统公共的部分而不仅仅是孤立的呢？如何使应用程序在作用域中成为共用的，而不是要依赖于单一的环境或语言？

15.1　集成Oracle产品

作为迈向开放式系统的第一步， Oracle Developer可以使用另外一个工具。在第4章已经看到了这种使用方法，即几个混合使用表单、报表和图形的基本方案。还有其他可能的组合，例如在一个图形显示中显示一个报表，或从一个报表运行一个表单，以及其他可能的建立集成的方法。还可以在一个饼图图表的一部分单击鼠标，产生有关该主题的一个报表 (虽然像这样的下钻操动作的技术细节已超出了这本书的范围)。

本节内容给出如何集成Oracle Developer产品的基础的概述。对于PL/SQL程序包以及如何在应用程序中使用这个程序包的细节，可以参考每个单独的产品的手册。 Oracle Developer还集成了Oracle Express的数据开采工具和Oracle Developer系统，同时允许将Oracle Developer的安装和使用与系统的要求和设计结合起来。

Oracle Developr的Forms组件有最宽广的工具阵列，以集成其他的 Oracle Developer产品。要问的第一个问题是：将要运行的系统是一个客户机 /服务器系统还是带有 Oracle Developer Server的一个三层(Three-tier)系统。客户机/服务器系统把Reports Runtime或Graphics Runtime当作访问数据库服务器的一个本地程序来运行。把运行程序的可执行部分安装在每一个运行报表和图表的客户机上。三层 (Three-tier)系统使用运行于中间层服务器上的 Forms服务器或Graphic服务器来运行报表或图形显示。因此，应用程序运行在哪种体系结构上，涉及到对其他Oracle开发工具的集成。

如果希望报表和图形显示透明地运行，而不管是客户机 /服务器还是三层结构，那么应该

China-pub.com 下载

使用Run_Product内部过程来运行该程序。如果只是作为客户机/服务器型的程序运行，则应该使用Run_Product或是Run_Report_Object。Run_Report_Object是一个与Form Builder的报表对象一起工作的内置子程序，用于集成表单和报表。如果运行在三层结构上，可以使用这两个子程序，但还可以使用Web.Show_Document内部过程，该内部过程允许把一个URL提交给中间层的Application Server，并返回一个在Web浏览器上显示的页面。下面几节说明如何使用这三个内置子程序把组件集成表单应用系统中。

也可以使用图表项从表单中运行图形。第4章的“在表单中创建图形显示”中给出了使用图表项建立图形显示的详细例子。

15.1.1 用Run_Product集成产品

Run_Product内部过程是在Oracle Developer Forms中使用两层方式运行其他Oracle产品的机构。

注意 在图形中，Run_Product过程是TOOLINT程序包的一部分，所以必须把程序包的名字加到过程名字上：TOOLINT.Run_Product。

下面是Run_Product内部过程的语法格式：

```
Run_Product(<product>, <module>, <communication mode>,
            <execution mode>, <location>, <list>, <display>);
```

表15-1给出这七个参数的可取值和含义。

表15-1 Run_product参数

参 数	可 取 值	描 述
<product >	FORMS、REPORTS、GRAPHICS、BOOK	要运行的Oracle Developer的组件：Forms、Reports、Graphics或Book
<module>	字符串	在运行产品时打开的模块或文档的名字
<communication mode>	SYNCHRONOUS、ASYNCHRONOUS	同步调用在退出所调用的产品以后将控制返回给Forms。异步调用立即将控制返回，并行运行所调用的产品
<execution mode>	BATCH、RUNTIME	调用一批产品，采用后台方式运行一个报表或显示，没有用户交互。运行时产品用前台方式调用运行产品，允许进行交互。只有Report和Graphics可以在批方式下运行
<location>	FILESYSTEM、DATABASE	这个参数通知本过程模块驻留的位置是在文件系统中还是在数据库中
<list>	Param-List对象	参数列表包含有传递给该产品的一系列参数，传递参数的名字或者参数对象的ID
<display>	字符串	这是图表项块的名字，其中将包含所请求的来自Graphics的显示

注意 当从表单向正被调用的产品传递一个记录组时，或者当希望在一个客户机的浏览器中查看HTML、HTML CSS或PDF的输出时，必须使用SYNCHRONOUS通信模式。当调用一个表单或者一个工作簿的文档时，必须使用RUNTIME执行方式。只有当产品是Oracle Developer的Graphics组件时，才使用display参数。

例如，当用户单击一个按钮时，可能想要运行一个报表。应该将下面的代码放到这个按

China-pub.com 下载

钮的When-Button-Pressed触发器中：

```
Run_Product(Reports, 'LedgerSummary', RUNTIME, FILESYSTEM,
            NULL, NULL);
```

提示 可以不用路径名来限定模块名。使用在中间层服务器上的注册变量REPORTS60_PATH，指明包含将要运行的报表的目录。因为不需要涉及模块，而仅仅涉及注册关键字，所以这样会很容易地实现将报表模块重新定位到其他目录上。

为了在三层结构中用 Run_Product来运行一个报表，必须要设置注册变量 REPORTS60_PATH(包含报表的目录列表)、FORMS60_OUTPUT(将要保留报表输出的目录)、FORMS60_MAPPING(FORMS60_OUTPUT目录的虚拟目录，例如 /report_output/)和FORMS60_REPFORMAT(根据是否要进行 Adobe Acrobat输出或者 HTML输出决定取“ PDF ”还是“ HTML ”)。

第10章的“ Run_Prodnct ”节中也包含有一些例子和补充解释，说明如何来使用Run_Product子程序的参数列表的性能。

15.1.2 用 Run_Report_Object 集成报表

内置子程序Run_Report_Object使用表单上的报表对象的设置信息，这个报表对象是在Form Builder中创建的。

首先创建这个报表对象并且设置好它的特性。然后在一个菜单项或一个表单触发器中的代码中使用这个内置子程序。下面的教程在一个报表对象中建立 LedgerSummary报表，然后编制When-Button-Pressed触发器的代码，来运行这个报表。

1) 在Form Builder Object Navigator中，选择Reports节点，并单击Add工具按钮。

Form Builder显示一个对话框，或是使用Report Builder建立一个新的报表，或者是输入一个报表的名字。

2) 输入名字“ LedgerSummary.RDF ”，或者使用“ Browse(浏览) ”按钮浏览找到Ledger-Summary.RDF 文件，然后单击OK(见图15-1)。

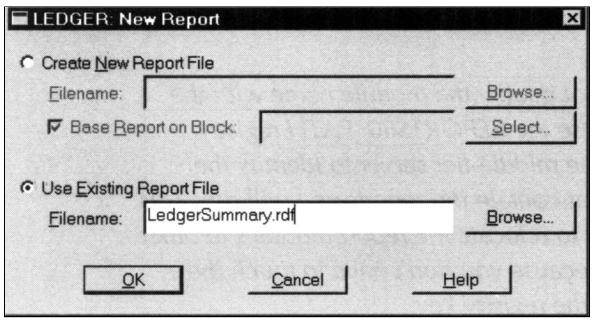


图15-1 输入文件名

3) 单击新的对象，改变新对象的名字，输入“ LedgerSummaryReport ”，再按回车键。

4) 双击该对象，显示Property Palette。

5) 设置Execution Mode属性为Runtime、Communication Mode属性为Synchronous、Report Destination Type为File、Report Destination Name为LedgerSummary.html、Report Destination Format为HTML、Report Server为repsvr.world(Reports Server的TNS名字)(见图15-2)。

China-pub.com
下载

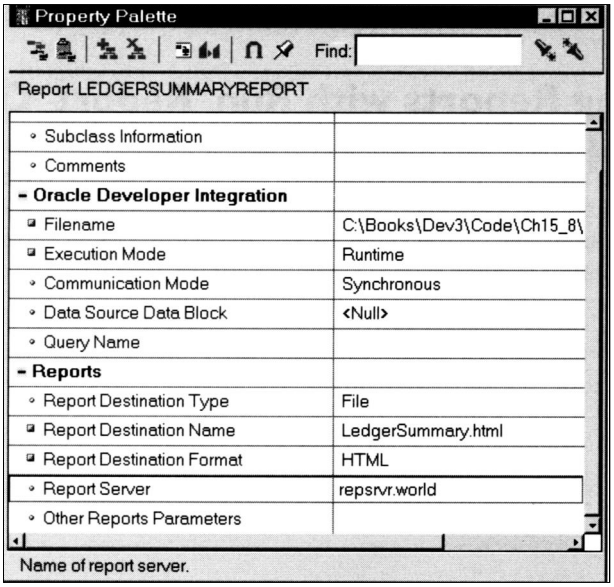


图15-2 Property Palette

提示 可以用一个命令行字符串形式在Other Report Parameters特性中提供任何的报表参数，例如用“PERSON_NAME=‘Adah Talbot’”来设置Person_Name参数。

6) 在画布上创建一个按钮项，并增加一个When-Button-Pressed触发器。

与Run_Report_Object、Report_Object_Status和Copy_Report_Object_Output过程一起使用这个报表对象，来设置和处理在服务器上或本地运行的一个报表。

```
DECLARE
  vReportID REPORT_OBJECT;
  vReportName VARCHAR2(100);
  vReportStatus VARCHAR2(20);
BEGIN
  vReportID := Find_Report_Object('LedgerSummaryReport');
  vReportName := Run_Report_Object(vReportID);
  vReportStatus := Report_Object_Status(vReportName);
  IF vReportStatus = 'FINISHED' THEN
    Message('Report ready.');
    Copy_Report_Object_Output(vReportName,
            'C:\TempReports\Ledger.htm');
  ELSIF vReportStatus IN ('RUNNING', 'OPENING_REPORT', 'ENQUEUED')
    THEN Message('Report running, please check for output later.');
  ELSE
    Message('Error when running report: '||vReportStatus||'.');
  END IF;
END;
```

Report_Object_Status函数可以返回这些值中的任何一个：finished、running、canceled、opening_report、enqueued、invalid_job、terminated_with_error、crashed。这些函数允许对报表进行处理，这些报表是在Reports Server上运行，而不是在当前进程中运行。可以等待报表的输出，可以报告状态并继续执行(如在例子的代码中所示)，可以定期地检查有关状态，使用报表的状态来告知用户何时可以使用报表。

China-pub.com 下载

15.1.3 使用Web.Show_Document集成Reports组件和Graphics组件

使用Web.Show_Document允许表单发送一个URL给一个Application Server。这个URL可以是任意的Web页的地址或者是要显示文件的地址，但是它也可以是一个 Cartridge或CGI命令字符串，该命令串通知服务器执行Report或Graphics的Runtime。

例如，如果在名为Ledger的表单上有一个用来运行名为LedgerSummary报表，并在Web浏览器中显示它的按钮，那么这个按钮的When-Button-Pressed触发器包含这样的代码：

```
Web.Show_Document(
  'http://www.talbotsfarm.com/reports/Reports60Cartridge?
  server=repsvr&report=LedgerSummary.rdf&userid=talbot@orcl&
  DESTYPE=CACHE&DESFORMAT=HTML', '_self');
```

引号中的字符串在程序单元中应该是在一个文本行上，在这里多行显示是由于长度的限制。这个命令的CGI描述与上面的描述非常相似：

```
Web.Show_Document('http://www.talbotsfarm.com/cgi-bin/rwcgi60.exe?
  server=repsvr&report=LedgerSummary.rdf&userid=talbot@orcl&
  DESTYPE=CACHE&DESFORMAT=HTML''_self');
```

CGI版本用CGI的虚拟目录代替报表的cartridge的虚拟目录，使用的是rwgi60.exe而不是cartridge的名字(Reports60Cartridge)。Web.Show_Document的第二个参数告知Forms Runtime在哪显示报表的输出：

- _SELF：在与表单相同的框架或窗口中装载报表。
- _PARENT：在表单的父窗口或框架中装载报表。如果该表单是最高层，_PARENT与_SELF相同。
- _TOP：把报表装载到包含该表单的窗口上，并替换表单的显示内容。
- _BLANK：打开一个新的、最高层的无名窗口，并将报表装入其中。

使用Web.Show_Document显示一个非常相似的图形：

```
Web.Show_Document(
  'http://www.talbotsfarm.com/graphics/Graphics60Cartridge?
  &module=AmountTrend.ogd&userid=talbot@orcl', '_self');
```

对一个图形的显示，要提供 MODULE和已经在Application Server上定义了的Oracle Graphics cartridge的名字。

15.2 通过ODBC使用其他数据库管理器

ODBC标准是一个用于数据库管理器的实际的标准应用程序编程接口 (API)。API接口使得所编写的程序可以从一种DBMS移植到另一个DBMS上，而不用重新编程，甚至不用重新编译代码。为了在一个非 Oracle的其他的DBMS上使用Oracle Developer，需要了解Oracle Developer如何能够通过ODBC来访问那种DBMS系统。

Oracle开放客户机适配器(OCA)是一个ODBC的接口，这个接口允许在Oracle Developer中使用ODBC的调用代替Oracle的调用。通过使用专门的数据库注册字符串，来访问这个软件层：

```
user_name/password@odbc:data_source
```

User_name和password是数据库的标准的名字和口令，data_source是数据库的ODBC的名字。data_source的前缀词‘odbc’告诉Oracle Developer要使用OCA接口。也可以用一个星号

China-pub.com
下载

作为data_source，在这种情况下，ODBC将用机器上已注册的数据源进行提示。

使用ODBC管理程序或者其他的数据库制造商提供的程序来设置一个数据源。有关设置的详细内容，查阅数据库管理程序文档或ODBC驱动程序文档。

注意 OCA接口支持各种ODBC兼容的数据源。Oracle带有几个具体的ODBC驱动程序和数据库的文档，但是OCA接口支持各种兼容的驱动程序。

Oracle提供了创建数据字典视图的脚本，这些数据字典视图与在 Oracle RDBMS中的类似。运行这些脚本允许使用Oracle Developer中的多种工具访问数据字典。可以创建自己的系统脚本，而不是Oracle所提供的那些脚本。只是复制一个脚本，然后根据所发现的系统具体的不同对它进行修改。

如果使用一个ODBC驱动程序通过OCA连接到目标数据库，有几个 Oracle Developer的功能将不能工作：

- 不能在数据库中存储模块。
- 不能访问Object Navigator的数据库对象组中的列的触发器的有关信息。
- 只能查看存储在数据库对象组中的过程，不能编辑它们。
- 不能在数据库对象组之间对过程进行拖动和放入。
- Oracle Developer不能使用约束信息产生主从关系或者强制主关键字。
- 如果在SQL语句中出现优化程序的提示，OCA不予理睬。

Oracle Developer有两种可用的工具，分别提供使用特定数据源的最佳使用方法。 OCA PL/SQL库函数包含有几个用于确定目标数据库是什么以及它提供哪些功能的子程序。EXEC_SQL库函数允许发出使用专用的语法说明访问服务器的 SQL语句。通常，PL/SQL代码必须遵循标准的Oracle PL/SQL的规定以及访问数据库服务器的特定语法。可以在ORACLE.INI文件中或者在系统注册表中，对OCA_DEBUG_SQL=TRUE和OCA_DEBUG_ERROR=TRUE进行设置，以便对将被发送到数据源的SQL语句进行调试。

Oracle Developer自动地根据情况对它的对象的特性进行调整，在需要时对它们进行修改，以求产生一致的结果。有一些确定的特性必须特别小心：

- 必须把与主关键字列相对应的项的 Primary Key特性设置为Yes，因为OCA不能从数据源得到这个信息。
- 如果得到不能更新主关键字列的有关错误信息，可能还必须把数据块特性 Update Changed Column设置为Yes。
- 如果试图一次要提取多于数据源能支持的记录，可能还得调整 Records Fetched这个数据块特性。
- 必须把数据源以混合形式存储的对象名用引号引起来。多数数据源对大小写不进行区分，但是有一些数据源，像Sybase，把“Ledger”和“ledger”区别对待。如果使用的是这些数据源，而且对象名字是用小写形式，那么在 SQL语句中必须用双引号(“)把名字引起来。

OCA提供了一个功能强大的工具，用于在非Oracle的数据源中的数据上打开应用程序。

15.3 将组件集成到Oracle Developer表单

组件是一个软件系统，可以与其他组件结合成为一个完整的运转的软件系统。组件一直

China-pub.com 下载

是被认为包含在一个表单或其他的对象中，但是随着面向对象的体系结构的广泛应用，这个概念具有了新的重要地位。Oracle Developer支持几种主要的组件标准。主要的两种标准是JavaBeans和ActiveX。下面两节分别地说明如何与Oracle Developer的Forms应用系统一起重新使用Java和ActiveX组件。

15.3.1 将Java组件集成到表单中

Java是一种编程语言，由Sun微系统公司不间断的和有争议的过程中发展起来，并形成标准。Java标准化的一个方向是JavaBean——一种组件说明，它允许在Java的小应用程序(运行在Web浏览器上的代码)、servlet(运行在Application Server上的代码)和application(运行在它自身系统上的代码)把Java类的系统当作组件使用。Java从C++程序设计语言中沿用了许多语法结构，进行许多改进和扩充。JavaBean说明使方法标准化，使它的系统的特性对客户机有效。

注意 如果想在应用程序中使用JavaBean创建自己的JavaBean类或者Java类，可能要求具备一定的Java的编程技能。例如，为了在一个bean区域中使用一个JavaBean，必须在Java中编写一个wrapper类，该类对由带有Developer服务器的Oracle提供的VBean类进行扩充(从它继承)。所有这些Java代码放入在应用服务器上的虚拟目录/codebase/中(有关这些虚拟目录的详细情况参看第14章)。由于讲授Java技能和方法超出了本介绍主题的范围，并且因为篇幅所限，这一节不会介绍集成到JavaBean需要编写的实际的Java代码。查阅Oracle Developer的帮助文件，了解可以使用的一些例子代码，仔细查阅随Oracle Developer系统销售的Oracle Developer演示的例子。

可以从第三方销售商处购买JavaBean，或者可以使用Oracle的Developer或是其他Java Developer编写自己的JavaBean。

可以通过两种方法来使用JavaBean：

- 链接：可以在一个数据块中创建专门的Bean Area项，使用Implementation Class特性链接到JavaBean，使用When-Custom-Item-Event触发器来处理在Bean Area中发生的事件。
- 定制：通过编写实现IView接口的JavaBean，可以用JavaBean来替代具体的控件(复选框、列表项、控制按钮、单选按钮组和文本项)，既可以直接地替换，也可以用扩展的VBean替代或者是一种其他的Oracle JavaBean类如VButton进行替换。

要把一个JavaBean控制链接到应用程序，在Form Builder中按下列步骤进行：

1) 打开将要显示控件的画布，单击Bean Area工具，拖画出将显示这个控件的一个矩形区域。

2) 编写扩充VBean类的一个包装类，例如像VCalendar(一个日历bean)。把Java和类文件放到在应用服务器上的虚拟目录/codebase/中，确保表单在运行时能够对它进行访问。

这个包装类提供一些特定的方法，Forms小应用程序调用这些方法来设置特性和获得鼠标事件。它还注册了表单能够识别的一些特殊事件。

3) 导航定位到在Object Navigator中的Bean Area项上，双击该项以显示它的Property Palette。将Implementation Class属性改为该包装类的名字。使用完全限定名字，包括整个软件包的名字，例如“oracle.forms”(见图15-3)。

图15-3 输入合法名字

China-pub.com 下载

4) 编写一个事件捕获过程并将它存储到一个程序库中。把这个程序库附加到使用JavaBean的表单上。这个过程从系统变量CUSTOM_ITEM_EVENT_PARAMETERS获取参数，从系统变量CUSTOM_ITEM_EVENT获取事件的消息。程序代码与下列内容相似：

```
PROCEDURE EventTrap IS
  vBean ITEM;
  vBeanParamList PARAMLIST;
  vParamType NUMBER;
  vEvent VARCHAR2(20);
BEGIN
  vBeanParamList := Get_Parameter_List(:SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS);
  vEvent := :SYSTEM.CUSTOM_ITEM_EVENT;
  vBean := Find_Item('CONTROL.CALENDAR'); -- gets the CALENDAR bean item from the
CONTROL block
-- Process the supported events registered through the wrapper Java class
END;
```

提示 可以在应用程序的程序单元部分编写这个过程。但是，当考虑可重用组件时，把该过程包装到可以在其他应用程序中重用的一个程序包中是一个好方法。或许想要有与JavaBean的集合相对应的一个库，这样，当使用一个bean时附加上这个程序库，同时事件捕获过程处于就绪和等待状态。

5) 从这个Bean Area项的一个When-Custom-Item-Event触发器中调用这个事件捕获过程：

```
EventTrap;
```

在PL/SQL代码中使用Set_Custom_Item_Property内部过程可以设置JavaBean对象中的属性，需要给出Bean Area项的名字，属性的名字（检查Bean文档）和属性的值。值可以是VARCHAR2、INT或BOOLEAN。

```
Set_Custom_Item_Property('Control.Calendar', 'enabled', TRUE);
```

开发一个标准的表单控件的定制版本，只需要替换在上述处理步骤中的 Bean Area控件。例如在一个文本项中，把Implementation Class属性指定为置换的JavaBean的名字，并且为这个文本项增加一个When-Custom-Item-Event触发器。

15.3.2 把ActiveX组件集成到Forms中

ActiveX是Microsoft专用的一种技术套件，它能够使Windows应用程序对各种各样系统所开发的不同方面使用标准的接口。每个ActiveX控件都是一个提供定义明确的接口（程序和图形接口）并能发送事件到客户端（例如应用程序或网络上的其他应用程序）的组件。ActiveX控件以前被当作是OCX控件，它是建立在组件对象模型（COM）标准上的，组件对象模型（COM）标准也是Microsoft的一项技术。

注意 只能把ActiveX控件用于在Microsoft Windows系统下运行的应用程序中。

通过在数据块中定义一个ActiveX控件的方法，把ActiveX控件链接到应用程序上。Oracle Developer导入该控件的方法和事件，创建与该控件进行接口的代码骨架，然后在已产生的过程内部编写代码。当该控件激活一个事件时，调用与该事件对应的过程。

注意 使用On-Dispatch-Event触发器，可以进行一些复杂的程序设计，允许在事件过程

China-pub.com 下载

是缺省限定过程的情况下进行工作(也就是说，不能调用导航的非限制的表单过程)。参看帮助文件，可了解有关这种高级ActiveX编程技术的更多信息。

要在应用程序中包含一个ActiveX组件，在Form Builder中执行下列步骤：

1) 选择“Program| Import OLE Library Interfaces”菜单项。Form Builder显示“Import OLE Library Interfaces”对话框(见图15-4)。

2) 滚动已登记的ActiveX组件，直到发现要查找的ActiveX组件(本教程使用OracleSpreadTable组件)。对话框中显示所有可用于这个组件的方法和事件 (见图15-5)。

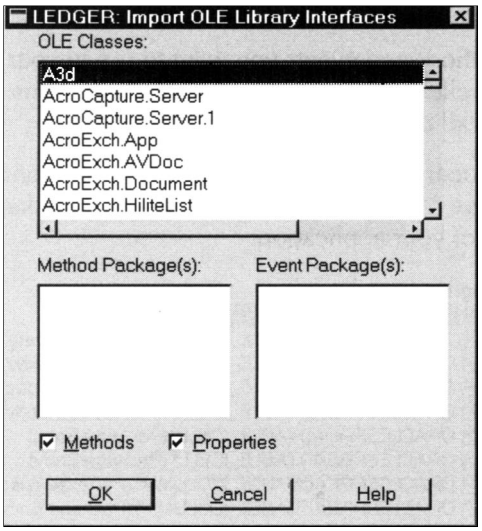


图15-4 Import OLE Library Interfaces对话框

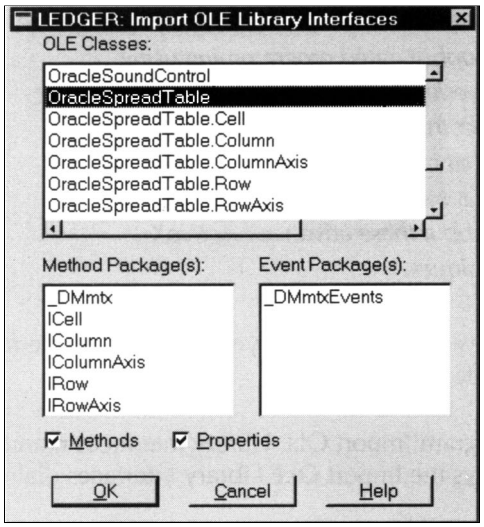


图15-5 对话框显示所有的方法和事件

3) 使用标准的选择多个项目的方法 (按住CTRL键，单击鼠标)，选择要在应用程序中使用的方法和事件。当选择了所有的方法和事件以后，单击OK。

然后Oracle Developer为所选择的方法和事件创建过程的骨架。在应用程序的 Program Unit(程序单元)段产生一系列的程序包 (见图15-6)。

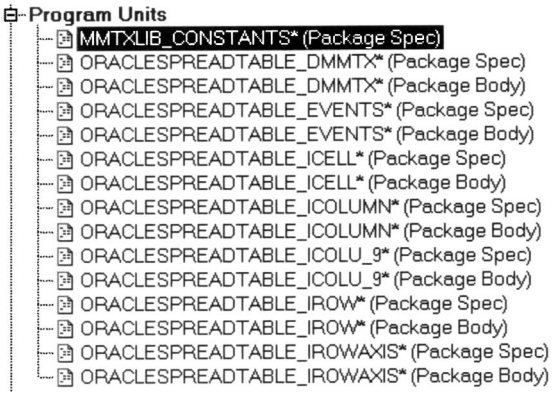


图15-6 Program Units段

4) 显示将要显示控件的画布。单击 ActiveX控件的工具，在要显示控件的画布上画一个矩形(见图15-7)。

5) 双击画布上的矩形控件，显示该控件的 Property Palette。单击“…”按钮，显示 OLE Classes的对话框。这是一个LOV，所以可以使用标准的LOV模式匹配想要查找的控件 (当在导入事件和方法时，在选择列表对话框中能找到相同的控件)(见图15-8)。

6) 选择控件名字，然后单击OK(见图15-9)。

7) 在画布的控件上单击右键，在弹出菜单中选择“Insert Object(插入对象)”。Form Builder显示Insert Object对话框。确保Create Control单选按钮被选定，然后单击OK(见图15-10)。

China-pub.com
下载

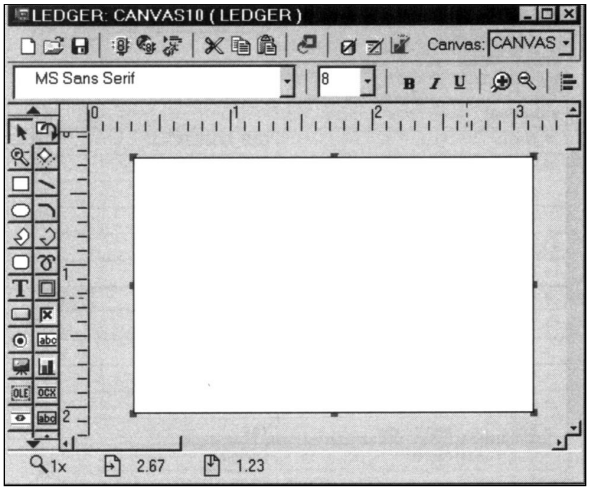


图15-7 在画布上画一个矩形

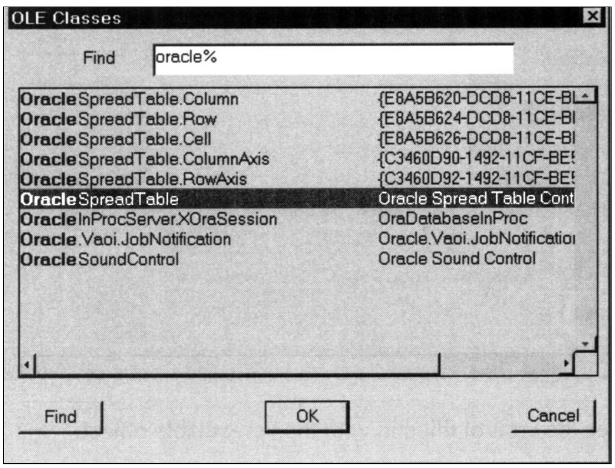


图15-8 OLE Classes对话框

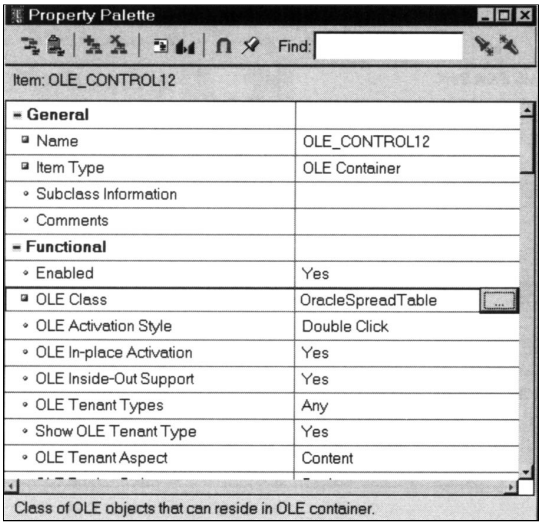


图15-9 选择控件名

China-pub.com 下载

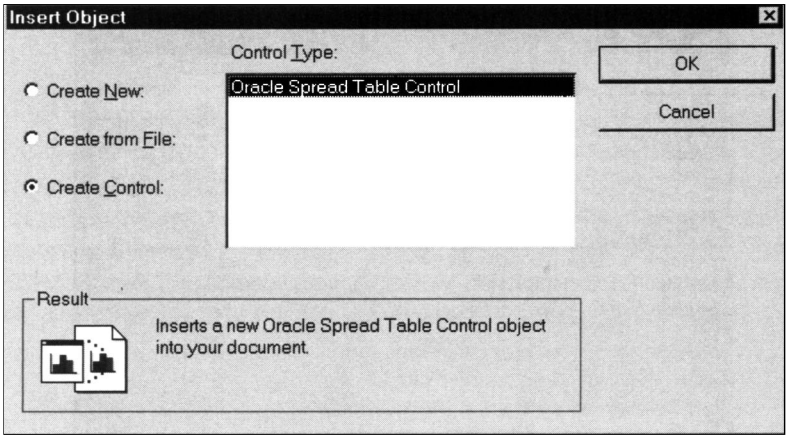


图15-10 在Insert Object对话框中选择Create Control单选按钮

然后会看到控件已被用Spreadtable对象填充(见图15-11)。

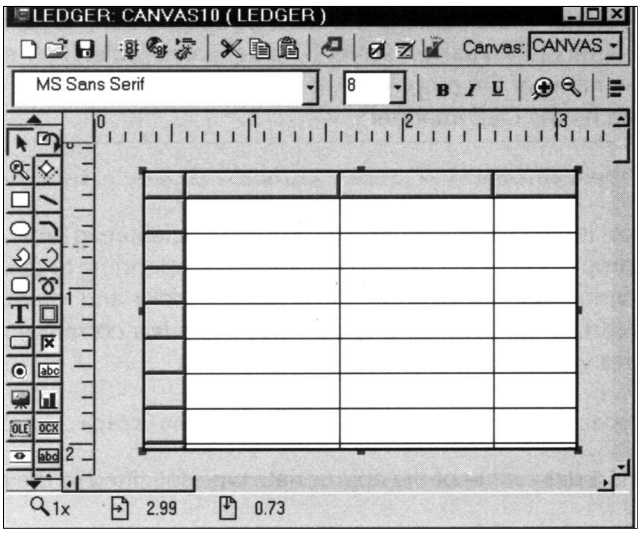


图15-11 填充Spreadtable对象的控件

提示 一旦已经“占据”了控件，就能够查阅有关该控件的任何文档，弄清楚每个事件和方法要做什么以及为了响应一个事件需要做些什么。通过控件的Property Palette，使用Control Help属性，可以访问内部文档。一旦已经设置了控件的名字，只需要单击“…”按钮即可。使用Control Properties属性同样可以设置控件的任何属性。

接下来需要进入到程序单元中的每个程序包，用实现已导入界面的代码来替换程序包中的NULL部分。也可以使用导入方法编写其他的子程序和程序包，做所想要做的各种事情。可以在PL/SQL中通过变量赋值语法，可以引用组件的属性：

```
vVar := :Item('<block>.<item>').ocx.<server>.<property>
```

其中<block>.<item>是数据块中的ActiveX控件项的名字，<server>是ActiveX服务器的名字(例如，OracleSpreadTable.1)，<property>是属性的名字。也可以使用由OLE导入程序生成的代码：

China-pub.com 下载


```
vVar := <package>.<property>(:Item('<block>.<item>').interface);
```

其中<package>是已生成的包的名字(例如OracleSpreadTable_lColumn)，<property>是对应于该属性的包函数(例如对列对象的位置特性)，<block>和<item>与前边一样，是数据块和ActiveX控件项的名字。有一个对应的用来设置值的过程语法：

```
<package>.<property>(:Item('<block>.<item>').interface, <value>);
```

其中<value>是在过程的参数表中说明的相应类型的数据值。

15.4 修改键值

这章最后的话题涉及到Oracle Developer Forms界面的一个小的但很重要的部分。Forms用户通过键盘上的键执行许多操作。可以控制这些键的赋值实现要做的任何事情。如果正在使用Developer Forms Server，在文本编辑器中编辑键资源文件。对于客户机/服务器的表单，Oracle Terminal允许修改用于执行命令的键，既可以在生成器中，也可以在Runtime程序中。

15.4.1 编辑Web键值

在安装Developer Server时，Oracle Installer增加安装一个称为FMRWeb.RES的资源文件：

```
#  FMRWEB.RES is the key definition file for webforms. The syntax is:
#
#    JFN : JMN : URKS : FFN : URFD   (whitespace ignored)
#
#      JFN = Java function number
#      JMN = Java modifiers number
#     URKS = User-readable key sequence (double-quoted)
#      FFN = Forms function number
#     URFD = User-readable function description (double-quoted)
#
#  JAVA FUNCTION NUMBER
#         33 = PageUp
#         34 = PageDown
#         35 = End
#         36 = Home
#         37 = LeftArrow
#         38 = UpArrow
#         39 = RightArrow
#         40 = DownArrow
#    65 - 90 = Ctrl+A thru Ctrl+Z (These will always have the control
#              modifier explicitly included, as well as any other
#              modifiers that might be used.)
#  112 - 123 = F1 thru F12
#          9 = Tab (Ctrl+I, without the control modifier)
#         10 = Return (Ctrl+J, without the control modifier)
#
#  JAVA MODIFIERS NUMBER
#  Equal to the sum of the values for the modifier keys:
#    0 = None
#    1 = Shift
#    2 = Control
#    4 = Meta
#    8 = Alt
#
#  FORMS FUNCTION NUMBER
```

China-pub.com 下载

```
#  The Forms function numbers match the function numbers found in a
#  typical Forms key binding file.
#
#  USER-READABLE STRINGS
#  The double-quoted strings appear when users click [Show Keys], and
#  are used for this purpose only. These strings can be translated as
#  needed. Note that the strings do not affect what actually happens
#  when end users press a particular key sequence.
#
9    : 0 : "Tab"             : 1  : "Next Field"
9    : 1 : "Shift+Tab"       : 2  : "Previous Field"
116  : 0 : "F5"              : 3  : "Clear Field"
38   : 0 : "Up"              : 6  : "Up"
40   : 0 : "Down"            : 7  : "Down"
33   : 0 : "PageUp"          : 12 : "Scroll Up"
34   : 0 : "PageDown"        : 13 : "Scroll Down"
69   : 2 : "Ctrl+E"          : 22 : "Edit"
10   : 0 : "Return"          : 27 : "Return"
76   : 2 : "Ctrl+L"          : 29 : "List of Values"
115  : 0 : "F4"              : 32 : "Exit"
75   : 2 : "Ctrl+K"          : 35 : "Show Keys"
83   : 2 : "Ctrl+S"          : 36 : "Commit"
118  : 1 : "Shift+F7"        : 61 : "Next Primary Key"
117  : 0 : "F6"              : 62 : "Clear Record"
38   : 2 : "Ctrl+Up"         : 63 : "Delete Record"
117  : 1 : "Shift+F6"        : 64 : "Duplicate Record"
40   : 2 : "Ctrl+Down"       : 65 : "Insert Record"
119  : 1 : "Shift+F8"        : 66 : "Next Set of Records"
1005 : 0 : "Down"            : 67 : "Next Record"
1004 : 0 : "Up"              : 68 : "Previous Record"
118  : 0 : "F7"              : 69 : "Clear Block"
66   : 2 : "Ctrl+B"          : 70 : "Block Menu"
34   : 1 : "Shift+PageDown"  : 71 : "Next Block"
33   : 1 : "Shift+PageUp"    : 72 : "Previous Block"
116  : 1 : "Shift+F5"        : 73 : "Duplicate Field"
119  : 0 : "F8"              : 74 : "Clear Form"
122  : 0 : "F11"             : 76 : "Enter Query"
122  : 2 : "Ctrl+F11"        : 77 : "Execute Query"
69   : 3 : "Shift+Ctrl+E"    : 78 : "Display Error"
80   : 2 : "Ctrl+P"          : 79 : "Print"
123  : 0 : "F12"             : 80 : "Count Query"
85   : 2 : "Ctrl+U"          : 81 : "Update Record"
121  : 3 : "Shift+Ctrl+F10"  : 82 : "Function 0"
112  : 3 : "Shift+Ctrl+F1"   : 83 : "Function 1"
113  : 3 : "Shift+Ctrl+F2"   : 84 : "Function 2"
114  : 3 : "Shift+Ctrl+F3"   : 85 : "Function 3"
115  : 3 : "Shift+Ctrl+F4"   : 86 : "Function 4"
116  : 3 : "Shift+Ctrl+F5"   : 87 : "Function 5"
117  : 3 : "Shift+Ctrl+F6"   : 88 : "Function 6"
118  : 3 : "Shift+Ctrl+F7"   : 89 : "Function 7"
119  : 3 : "Shift+Ctrl+F8"   : 90 : "Function 8"
120  : 3 : "Shift+Ctrl+F9"   : 91 : "Function 9"
```

使用任何一种编辑器，例如 Notepad，可以编辑这些代码和键的描述，给任何键分配任何功能。可以对这个文件中的任何一个键重新映射，改变第一列 (Java功能代号)和第二列(Java的限定符代号)中的代码，这两列合起来建立完整的键的设置。在靠近这个文件头的上边代码中

China-pub.com
下载

有详细的定义。不要改变 Forms功能代号(第4列)。可以通过对两个字符串的改变来修改在键的帮助屏幕上出现的内容，第一个字符串是键的描述，第二个字符串是功能描述。

15.4.2 用Oracle Terminal指定键

作为一个客户机/服务器的应用程序，运行的Oracle Developer Forms组件从一个资源文件中获取它的键的指定情况，例如从 Windows中的FMRUSW.RES文件。Oracle Terminal程序允许对那个文件的部分内容进行编辑，定义一个键的赋值。可以把键与应用程序的功能或键的触发器结合在一起，这样允许修改表单应用程序的键盘接口，以适应实际需要。也可以修改Form Builder的接口，以适应自己的任务。

注意 如果是通过一个Web浏览器运行应用程序，则不能够用Oracle Terminal改变键的赋值。

1) 在运行Oracle Terminal时，它给出一个文件提示。找到 Forms的资源文件 (通常是fmrusw.res)，单击OK。在Windows上，该文件是在FORMS60目录中。如果试图使用一些其他的资源文件，将会发现这些文件不具备用于 Oracle Terminal的正确的结构。然后Terminal显示主终端窗口(见图15-12)。

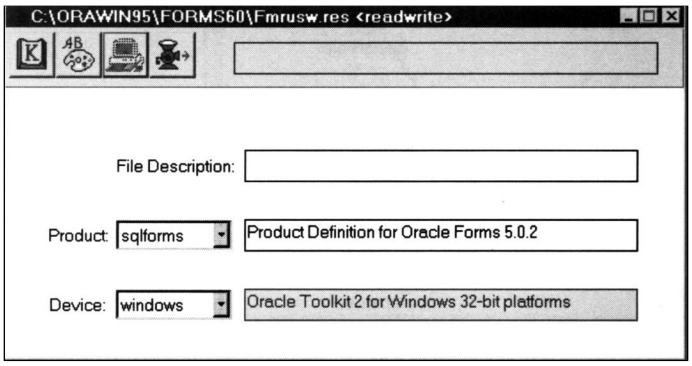


图15-12 主终端窗口

如果文件定义的产品和设备不止一个，这个窗口允许设置产品和设备，这对当前产品并不适用。

2) 单击Key Binding工具，利用Key Binding Editor(键赋值编辑器)对键的值进行编辑。

Key Binding Editor显示键分类树(见图15-13)。每一种分类都包含一个键赋值集合，与一种特定的操作类型相关联。中间级分类 (runform和design)具备可应用于该产品中的所有情况的操作，例如Show Keys、Help、Cancel和Exit。叶分类具备应用于特定情况的操作。

3) 为了看到键的赋值，双击分类的圆圈。

编辑器显示一个赋值表 (图15-14显示的是“ normal ”分类节点)。这个表给出了“ 操作 ”，如[Next Block]或[Scroll Up]，以及对其进行赋值的键，如CONTROL+D或SHIFT+F9。对一个操作可以有多行，指示可以通过任何一组键进行该操作。对一个操作的每一个键在表中要占据一个独立的行。

4) 要修改一个赋值，查找到该操作，用键入的内容覆盖该键。

可以用Row按钮(Duplicate Row、Insert Row、Delete Row)对行进行操作。也可以输入键，

China-pub.com 下载

把游标放入键的区域中，单击Macro Mode按钮。然后可以输入该键，按下真正的键，而不是拼写出单词如 ” Control ” 和 ” Shift ”。按照结果对话框的指导，完成宏的录制。

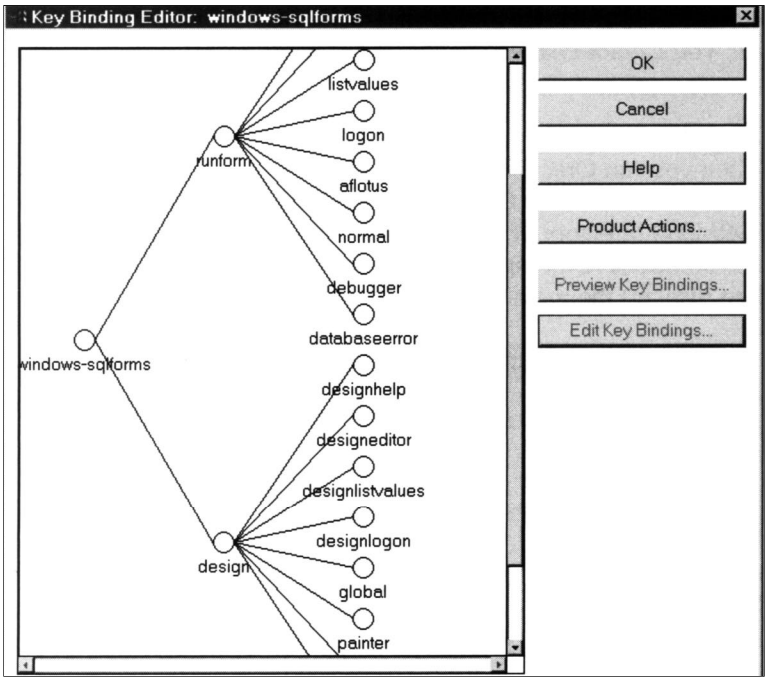


图15-13 Key Binding Editor

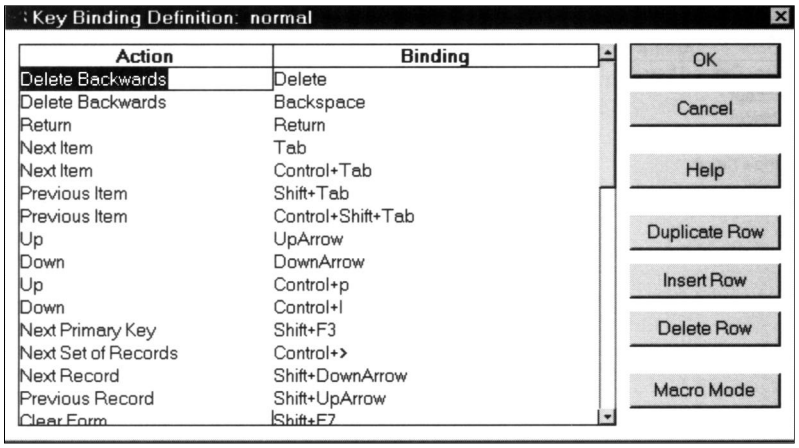


图15-14 Key Binding Definition Editor

设置键的触发器是一个更复杂的过程，因为需要定义操作。要在键赋值编辑器中的最高层windows-sqlforms上进行这项工作。

5) 双击windows-sqlforms节点，看到一个空的操作表。

6) 单击Insert Row按钮插入一行。

7) 插入对应于键触发器的操作(例如，KEY-F3)。然后如前所述，输入该键的赋值。

8) 现在单击Product Actions按钮，调用Product Action Editor，它的图像与Key Binding Editor非常相似。

China-pub.com 下载

9) 双击sqlforms分类，看到Product Action Definition窗口(图15-15)。

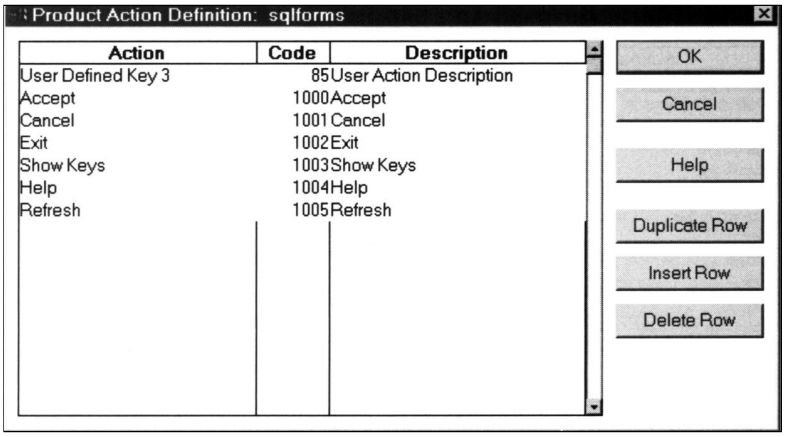


图15-15 Product Action Definition窗口

10) 输入操作(例如，[User Defined Key 3])、它对应的数值编码(从82到91对应‘0’键到‘9’键，在这种情况中编码(85)对应‘3’键)、以及显示在键赋值列表中的描述。单击OK。

11) 当完成定义赋值时，单击OK，释放键赋值编辑器，返回主窗口。

12) 保存键的赋值。

13) 单击Generate工具，生成与新的赋值对应的新的资源文件。

14) 再次保存，完成该过程。

15.5 用Oracle Translation Builder使应用程序国际化

由于因特网的出现，世界正变得越来越小。作为一个经验法则，如果从开始对应用程序就提出了语言可移植方面的要求，将会处于主动。进入 90年代，世界已经变得非常地小。如果开始就知道必须使应用程序支持挪威语、俄语、阿拉伯语或者是汉语，就有可能设计出能够适应这些语言的应用程序，而不是去改变语言的要求。

为了便于把Oracle Developer的应用程序翻译成不同的语言，需要了解三个方面的内容：

- 本国语言支持(NLS)。
- 版面设计。
- 字符串转译。

15.5.1 Oracle中的本国语言支持

Oracle提供一种扩充功能，称为本国语言支持(NLS)，用于处理语言方面的差异。在《Oracle 8i数据库管理员手册》中可以了解这个功能的详细信息。如果将要在多种字符集和语言环境中工作，应该先从阅读Oracle文档中有关NLS的内容着手，因为应该清楚地知道Oracle能干些什么。

NLS提供对这些语言方面的支持：

- 存储在数据库中的、不同语言的文本和字符集。
- 来自Oracle和运行时程序的不同语言的消息。
- 对不同的语言相应格式的默认掩码。

China-pub.com 下载

■ 在SQL表达式和ORDER BY子句中使用关系操作符的不同字母表的排序和整理顺序。

■ 在表单、报表和图形中被显示的项的双向格式化。

在这种情况下，所谓“支持”，意味着Oracle透明地处理各种事情。作为一个应用程序开发人员，除了告诉Oracle希望使用的是哪种语言与地域以外，不需要做任何事情。在系统注册表中用NLS_LANG环境变量说明语言种类。

可以使用ALTER SESSION语句(Oracle SQL对ANSI标准的一种扩充)修改NLS的任何参数。如果愿意的话，可以练习一下控件对语言支持的敏感程度。本书建议对此少花力气，既是因为NLS功能并不符合已形成的ANSI标准，又是因为对于这种类型特性通常最好接受它的缺省值。不然，可能要不断地开足马力进行测试、调整和研究，才能得到正好是利比亚的阿拉伯语的方法，或者可能根据流程继续进行…

另一个根据流程继续进行的地方是使用缺省的格式掩码。所有的 Oracle Developer的产品都使用格式掩码来格式化数值型和日期型数据。本书中的各种例子建议用设置格式掩码的方法来说明日期型、数值型的特定值和货币值。如果将来提供的是基于语言的应用程序，可以不采用这个建议。使用缺省的格式掩码， NLS将自动地使用与在NLS_LANG中说明的语言合适的掩码。如果使用的是显式格式，则将失去自动处理功能。然后必须对所包括的每个事情进行编码，众所周知，这是很乏味的工作。

有几个特殊的可以使用的格式掩码字符， Oracle使用NLS能正确地对它进行翻译：

字符	含义
C	国际货币符号
L	本地货币符号
D	数字中的小数点
G	数字中的千位分隔符

例如，L999G999D99是$999,999.99的一种国际上通用的写法。 NLS也会把不同的日期组件翻译成本地的写法，例如月份名或者星期几。参看 Oracle Developer中Forms、Reports、Graphs等组件的联机文档，了解有关格式掩码的详细内容。

有一种特殊情况必须注意：PL/SQL表达式。如果在PL/SQL中因某种原因使用日期常量，应把它包含在一个To_Date函数中，以确保PL/SQL能够正确地处理它。例如，这里是一个依赖语言的表达式：

```
:Ledger.ActionDate := '1/1/01';
```

它依赖于语言，因为PL/SQL使用标准“美国式”格式掩码“dd-MON-yy”对该字符串进行转换，而不去管当前使用的语言。应该使用这样的代码去代替：

```
:Ledger.ActionDate := To_Date('01/01/01', 'DD/MM/YY');
```

15.5.2 布局设计

布局设计中关键的问题是基于窗口系统的可移植性问题：必须在不同时刻调整对象的不同的大小。对于一个基于窗口的系统，因为缺省字体的作用，按钮的大小是不同的。在一个系统中某些按钮是双线边框，而在另一个系统中又没有。

对于不同的语言，有几个布局问题：

■ 文本字符串的大小随被转换的字符串的长度而变化。

China-pub.com
下载

■ 格式化过的字符串(格式掩码区)的大小随语言而变化。
■ 界面组件“ widgets ”的大小随语言而变化。界面组件是显示在屏幕上包含有文本信息的各种图形对象，比如使用各种语言的单选按钮。
■ 标签的位置随它们所标注的区域而变化，均依赖于语言。

对所有这些情况，Oracle建议使用一个经验值，即预留30%的扩展位置。如果能这样做，可以开发出这种语言占据大部分位置的原始的应用程序。

同样地，可以能设计出具有语言可移植性的屏幕、报表和图形的布局。尽量少使用样板，这种静态的资料将会在需要扩充区域时，增加要求转换的文本的每一位而得到。不要与图形一起混用模板，那样将会妨碍进行扩展。重新编排区域，这样不同的语言将显得是合理的，即使在所开发的应用程序中语言的阅读方向是反读的。

15.5.3 字符串的转换

开发的任何应用程序都将有大量的字符串。为了使字符串的转换减少到最低程度，应该在模板中尽可能少地使用字符串。例如，对需要成为国际上通用的应用程序，不要把大量的要使用的文本指导信息放到模板上。可以把指导信息放到数据库中，并在一个只读的区域中显示它们。

在Oracle Developer中有两种类型的字符串值得担心：

■ 模板文本，包含有标签和标题。
■ 在PL/SQL过程中的字符串常量和变量。

1. 模板文本

模板文本是在表单、报表或图形中的一种文本，这些文本是通过一个编辑器输入的，而不是从数据库填写的。设计者将这个文本用不同的格式保存到应用程序文件中 (或者是数据库中，如果数据库存在)。

模板文本的转换是一个查找该文本并将它翻译为所希望出现的语言的过程。至少有两三个方法来研究这个问题。首先，可以制作出对各种语言的应用程序文件 (表单、报表、图形)。其次，可以用显示区域来代替模板，并从数据库进行填写，用专门的 PL/SQL代码变换出合适的语言和数据的位置。本书建议使用前面的方案。维护应用程序的数据，和维护那些不得不增加到应用程序中的相对复杂的代码相比起来，要简单的多。

再次说明，虽然一味地减少文本是很困难的，但是限制这样文本的数量对国际化的应用程序而言仍是一个很好的策略。

2. Oracle Translation Builder

Oracle Translation Builder的工作是把在一个Oracle Developer的应用程序从一种基本语言翻译成为多种目标语言。它通过在数据库服务器上建立的一系列的数据库表进行这项工作。

要使用Oracle Translation Builder，首先必须把它的表安装到数据库中 (参见附录B有关安装数据库表的细节)。一旦已经安装了数据库，从任务条上启动 Translation Builder。

1) 选择Start| Oracle Developer 6.0|Translation Builder。这个菜单项启动该Builder，Builder显示一个初始化屏幕，提示选择是在一个工程中工作，还是运行 Quick Tour。选择在一个工程中工作(见图15-16)。

2) 第一个任务是建立一个标准的连接，从主菜单上选择File |New |DB Connection。图15-17

China-pub.com 下载

给出建立了Talbot连接的Navigator。

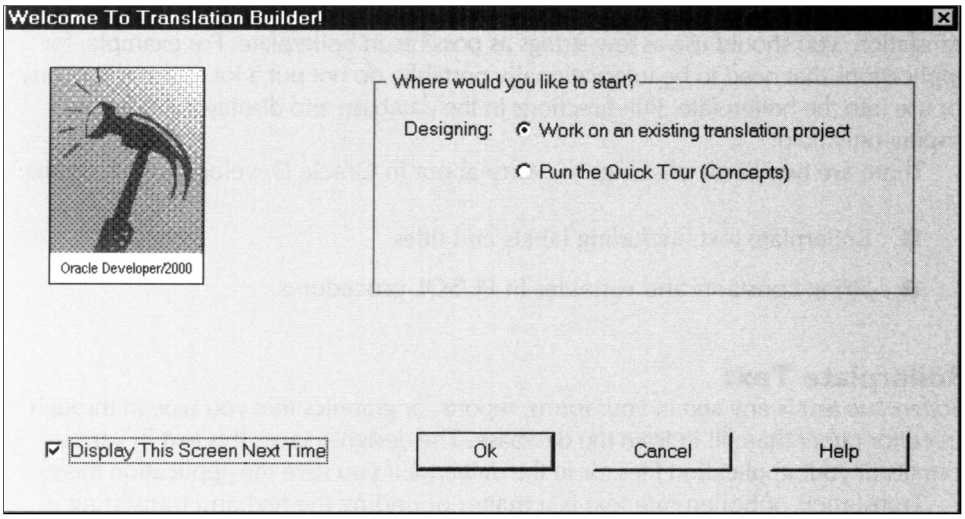


图15-16 选择在一个工程中工作

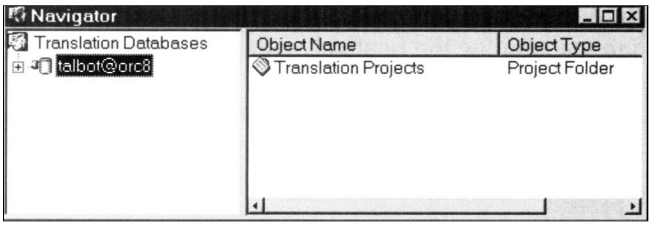


图15-17 Navigator窗口

3) 现在需要创建一个工程文件。这个工程文件对应于要捆绑到一个单个的应用程序中的一系列模块，或者是对应于其他的希望作为一个系统看待的 Forms模块组。单击 File |New |Project，在数据库连接之下创建一个工程。出现 New Project(新工程)对话框(见图15-18)。

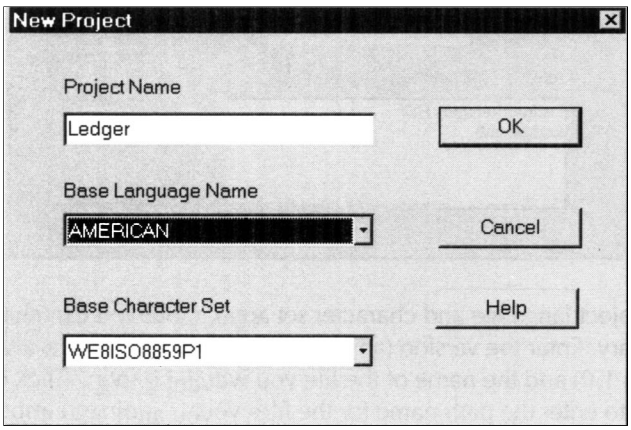


图15-18 New Project对话框

4) 要创建一个工程，必须指定一个名字和基本语言以及字符集。对于后两项，最佳的选择是NLS_LANG环境变量的当前设置值， NLS_LANG环境变量可以从与所有其他的 Oracle变

China-pub.com
下载

量一起的System Registry中找到。例如，在本例中，Talbot使用美国语言和WE8ISO8859P1字符集。单击OK，将会看到新的工程出现在工程文件夹中(见图15-19)。

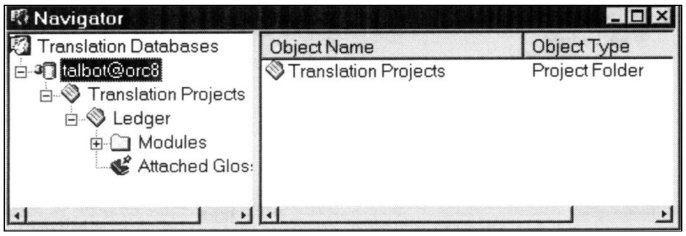


图15-19 工程文件夹中的新工程

5) 创建转换的第一个步骤是把基本版本引入到工程中。打开该工程，然后单击 Module |Import Strings菜单项或者 Import Strings 工具，将会看到这个Import Strings对话框(见图15-20)。

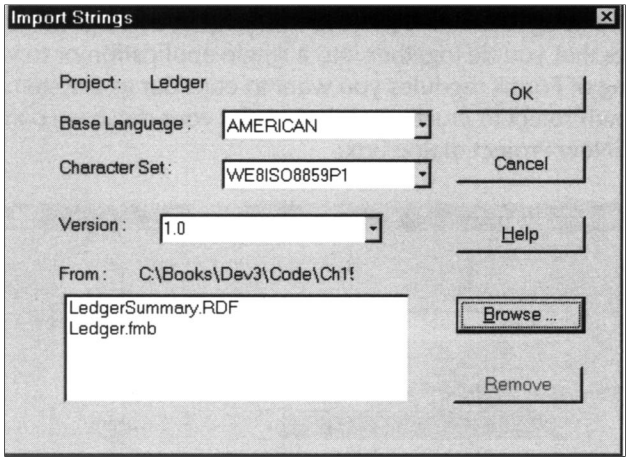


图15-20 Import Strings对话框

6) 出现工程的语言和字符集，但是如果需要可以修改它们。输入版本 (任何字符串，但是最佳的选择是一个版本号比如 1.0)和希望引入的文件的名字。单击 Browse按钮，输入将要引入的文件的路径名，并从出现的 Open File对话框中选择文件 (这里是Ledger的表单模块和LedgerSummary的报表模块)。会看到已选择的文件列表和那些文件的路径。单击 OK结束。Navigator(导航器)现在就包含有新的模块，如图 15-21所示，并可以开始进行转换。

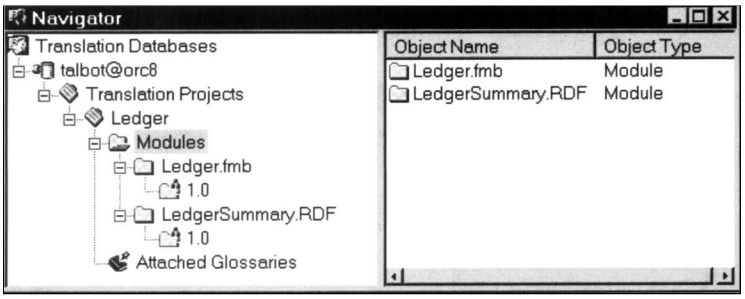


图15-21 包含有新模块的Navigator(导航器)

7) 打开一个模块的版本文件夹。用菜单项 Module |Add 或Add Translation工具条来增加

China-pub.com 下载

一个转换模块。显示Add Translation对话框(见图15-22)。

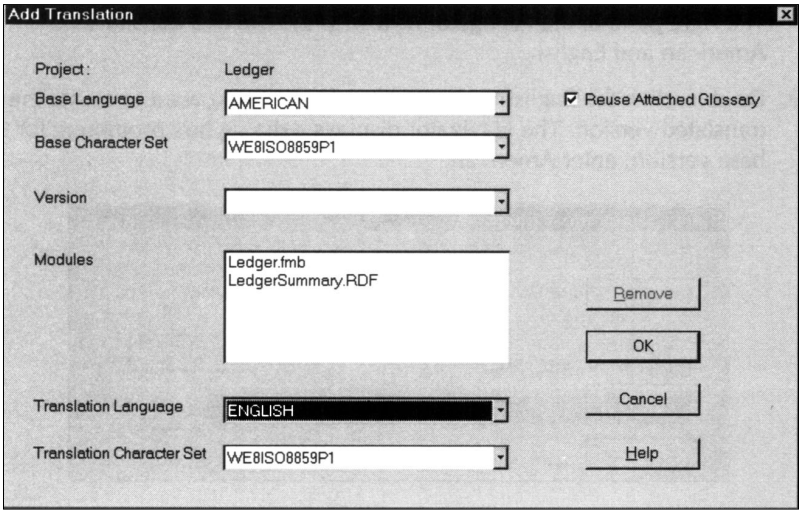


图15-22 Add Translation对话框

8) 在本例中， Talbot进行一个英式英语的转换。改变语言，由美语变成了英语。拉下Translation Language列表框并选择English(英语)。字符集保持不变。单击OK创建新的转换，它出现在Navigator(导航器)中的版本下面(见图15-23)。

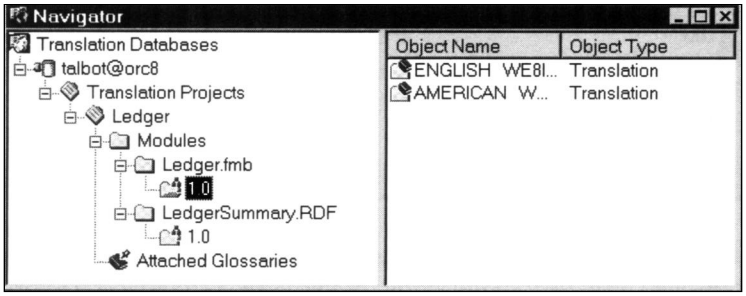


图15-23 Navigator下新的转换

注意 查阅有关Oracle的NLS特性的文档资料，了解语言和字符集的一个完整的列表。

现在在Navigator右边的显示这两个转换器：美语和英语。

9) 双击英语转换器，指定要创建该转换的版本。 Navigator(导航器)显示一个对话框，提示输入基本版本，输入美语(见图15-24)。

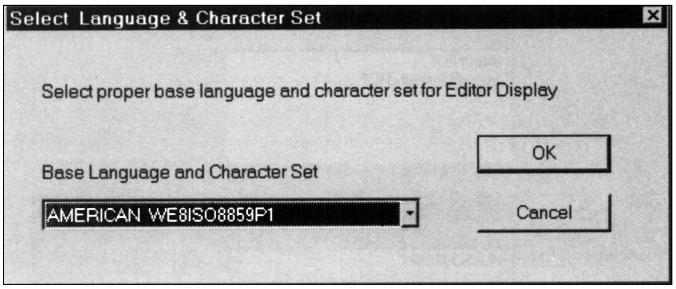


图15-24 在对话框中输入美语

China-pub.com
下载

现在会看到Translation Editor(转换编辑器)，有多排的字符串列表(见图15-25)。

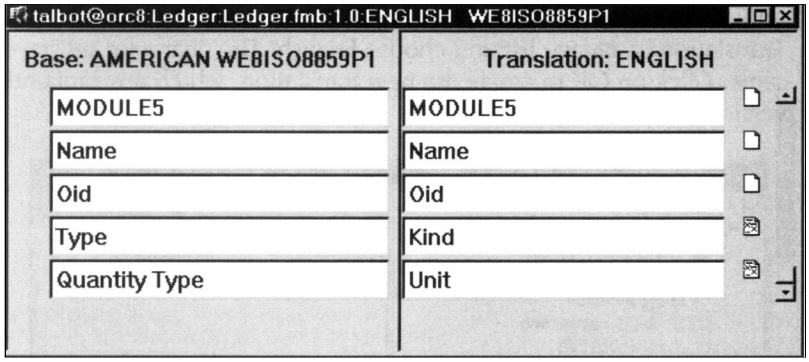


图15-25 多排字符串列表

10) 改变需要转换的字符串。对改变的字符串，会看到已转换图标显示在字符串旁边。在本例中，Talbot已经把Type转换为Kind，把Quantity Type转换为Unit，而这些词语将非常容易地被英国用户认识。当完成了编辑时，单击 Save工具，然后在Editor中的标题条中用关闭窗口控制按钮关闭该Editor。

接下来，进行最后一步——把转换集成到原来的模块中。

11) 在Navigator中选择Ledger节点。单击Module |Export Strings菜单项或者是Export Strings工具条。会看到Export Strings对话框，类似于Import Strings对话框(见图15-26)。

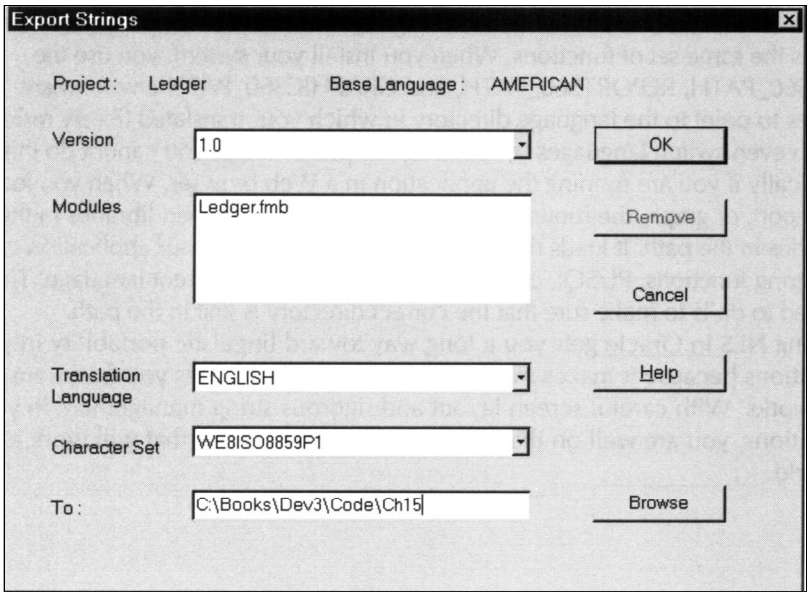


图15-26 Export Strings对话框

12) 用Remove按钮选择并删除不希望导出的任意模块。在本例中，Talbot已经删除了LedgerSummary报表，因为它没有被转换过。

13) 从下拉列表中选择导出的版本。输入要导出的转换的语言和字符集 (英语，被转换的语言)。单击OK。

China-pub.com 下载

转换生成器把转换结果写到模块文件中，现在模块文件包含有两种字符串的集合。当运行于合适的NLS_LANG设置时，看到的是转换后的字符串，而不是基本的字符串。

3. PL/SQL字符串

在许多时候，PL/SQL代码将通过警告或其他机制显示错误、警告或劝告信息。经常是通过自动地并置所生成的字符串来构造这些信息。如果希望制作一个国际通用的应用程序，不得不采取一些方法保证这些字符串以正确的语言出现。可以把它们进行编码，动态地检查当前的语言和回答的正确性，但是有一些比较好的方法可以达到同样的效果。

最有效的方法是把字符串集中到一个 PL/SQL库中进行处理。可以进行简单地编码，把信息与要查找的信息编号固定，或者可以开发专用函数，进行参数化的输入并产生一个合适的信息。为每一种希望支持的语言建立一个库，每个库包含同样的函数设置。当安装系统时，使用FORMS60_PATH、REPORTS60_PATH和GRAPHICS60_PATH环境变量，指出被转换库所驻留的语言目录。甚至可以通过重置这个路径对语言进行转换。可是如果在一个 Web浏览器中运行应用程序，则不能动态地这样做。当装载一个表单、报表或图形时，运行中的程序在路径的目录中查找附加的库。它装载所发现的第一个版本。当应用程序调用一个字符串函数时，PL/SQL调用为当前语言所定义的函数。这样，需要做的所有事情就是确信正确的目录是路径中的第一个目录。

使用Oracle中的NLS能够获得一个在应用程序中面向语言可移植的长远的方法，因为它使得许多字符集问题变为透明的：只是进行操作。在应用程序中严谨的屏幕布局和严格的字符串管理，将会使你的面对国际环境的应用程序取得成功。

China-pub.com
下载

第五部分　其　　他

第16章　对象和属性参照表

本章给出了Oracle Developer中的所有对象的一个完全索引表。Oracle Developer环境有三个主要组件：Form Builder、Report Builder和Graphics Builder，每个组件各占一部分。

16.1　Form Builder

Form Builder集成了几个模块：表单、菜单、PL/SQL库和对象库。这两个库模块不再包含任何的内部对象。Form Builder还包括一系列的内嵌子程序，这些子程序在Object Navigator中使用，在内嵌程序包中介绍。Report Builder部分列出了数据库对象，给出了访问数据库服务器数据字典的方法(用户、表、列、编译单元和触发子)。Object Navigator节点如何进行工作的说明，参看第2章。在第10章的“编写一次代码”中，介绍如何利用Database Objects节点去定义编译单元并将编译单元从客户机移到服务器上的内容。

16.1.1　表单模块

图16-1用图的形式给出了表单模块结构的概述。第2章的“表单模块”介绍了这个结构和表单模块对象的功能。

1. 表单模块的对象

表16-1列出了表单模块的对象，并描述了它们在表单应用程序中的功能。这些对象是嵌套于图16-1中主对象的对象。表16-1列出了所有在表16-2中有属性的对象，不管它们是否直接地出现在图16-1或在Object Navigator中。为方便检索，表16-1中的对象是按字母顺序排列，而不是按它们在Object Navigator中出现的顺序排列。

表16-1　表单模块对象

对　象	描　述
Alert	显示一个信息或提出一个简单问题的标准窗口，用户根据所提供的按钮回答是/否，按钮最多有三个：Yes、No或Cancel
Arc	由曲线组成的图形对象，嵌套在画布对象中
Attached Library	附属到另外一个模块的PL/SQL库，使用库的目的是用变量代替程序单元
Canvas	一个窗口的背景，可以把样本文本和项放在其上
Data Block	拥有一个记录集的对象，每个记录由一系列项构成
Editor	显示为一个文本编辑器窗口的对象，用于把文本输入到文本项中
form	按照联机格式显示数据的一个模块，由一系列项组成，这些项被安排在一个或多个窗口中
Frame	画布中的一个对象，包含作为一个整体的一组块项的布局属性
Graphic	画布的图形元素(例如，线，等等)，嵌套在画布对象中
Image	一个矩形的图像图形元素，嵌套在画布对象中
Line	一种画布上的图形对象，嵌套在图形对象中

China-pub.com 下载

(续)

对 象	描 述
LOV	与选择列表一样，显示为一个专用对话框的一种对象，可以从一个值的列表中选择一个单一值。可以把一个LOV与一个记录组进行关联，也可以与一个项进行关联，以检查输入到该项数据的有效性。它也有搜索功能，容易地从一个大的列表中查找到某些项
Object Group	为在以后进行复制或者引用，对可以重复使用的某些对象进行包装的一种对象。它把一些对象集中在一个专门的标题下
Parameter	通过模块的运行系统将所分配的值传送到模块中的一种对象。该对象可根据模块实际所用的值来运行，以提高模块在不同的情况下的可重用性。参见系统参数和用户自定义参数
Popup Menu	一个移动的菜单，在画布或项上单击鼠标右键时弹出
Program Unit	一个PL/SQL的程序包，函数或过程
Property Class	集中了所有类型属性的一个对象
Record Group	表示一个专门的数据结构的一种对象，类似一个表，带有行和列
Relation	表示一个主从关系的一种对象。依据该对象明确两个数据块之间的映射关系，嵌套在关系中的主数据块中
Report	是表单与一个Oracle Developer Reports模块接口的一种对象。允许表单通过调用 Run_Report_Object内部子程序来运行报表模块。可以其他方法运行报表（调用Run_Product和Web.Show_Document内部子程序）
Tab Page	在一个选定画布上的一个独立的标签布局
Text	嵌套在画布下的一个图形对象，表示在该画布上的样本文本
Trigger	附加在另一个对象上的一个无名的PL/SQL块，在发生与那个对象有关的特殊事件时引发执行。触发器的名字反映出触发PL/SQL块的事件的类型
Visual Attribute	集中了可以从另一个对象引用的一批显示属性的一种对象。该对象操纵几个独立的属性：字体名字、字体大小、字体字形、字体间距、字体单位、前景颜色、背景颜色、填充模式、字符模式逻辑属性和黑白效果
Window	定义应用程序显示的一个矩形区域的一种对象，由 GUI平台提供

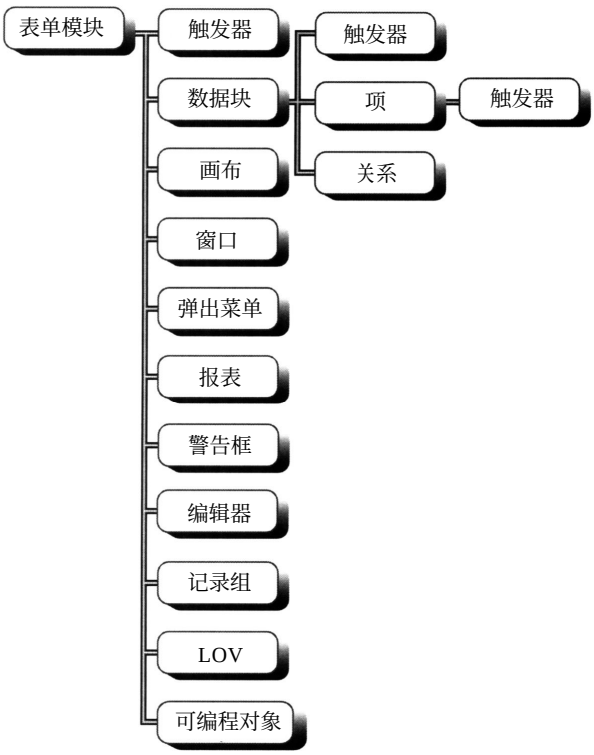


图16-1 一个表单模块的对象层次结构

China-pub.com
下载

2. 表单模块的属性

表16-2按字母顺序列出了表单模块的所有属性。描述属性的同时，表中给出了使用每个属性的对象。如果属性的值是有限的，描述部分列出了可能值。如果看见“是否”一词，意味着该属性是布尔型属性，只能选择Yes和No。Get/Set列说明是否可以使用适当的Get或Set内嵌子程序(Get_Form_Property、Set_Item_Property,等等)对该属性进行操作。“G”是指只能读取，“S”表示只能设置，而“GS”表示读取和设置都可以。“ ”表示不能在程序中对该属性进行操作。

表16-2 表单模块的属性

属 性	对 象	描 述	Get/Set
About Control	ActiveX 控制项	显示ActiveX控制的有关屏幕，只在通过插入一个对象而占据该控制之后可用	
Access Key	按钮、单选按钮、复选框项	表示可以用于选择一个项目的键的单个字符，与ALT(或COMMAND)键一起联用，等同于用鼠标单击此项目	
Alert Style	警告框	可取值为：Stop、Caution或Note。指定在信息旁边显示警告的图标	
Alias	数据块	对应于数据块的表的SQL别名。对于带有REF或对象类型的列的Oracle 8表，当需要在WHERE短语或ORDER BY短语属性中引用数据列时，要使用本属性。用Data Block Wizard自动设置	
Allow Empty Branches	层次树	确定分支节点是否可以没有子分支节点：缺省值为No，表示Form Builder只对分支节点进行转换而不包含它的子分支节点及叶节点。而Yes表示运行状态的表单把一个空节点当成一个被折叠的节点进行显示	
Allow Expansion	帧	确定Form Builder是否可以在帧的内容变化超过了当前边界时对帧进行扩展。缺省值为No，一般把它改为Yes	
Allow Multi-Line Prompts	帧	确定Form Builder是否可以把提示信息拆分为多行，以保留某些空白。缺省值为No	
Allow Star-Attached Prompts	帧	确定在表格式样的帧中是否在有位置时把提示信息放到起始位置而不是顶部，缺省值为No	
Allow Top-Attached Prompts	帧	确定在表格式样的帧中是否在有位置时把提示信息放到顶部位置而不是起始位置，缺省值为No	
Application Instance	表单、数据块、项	不是输入的任何信息，本属性包含对Microsoft Windows应用程序例子的引用(NULL表示其他平台)，可以用过程Get_Application_Property进行访问。可以把这个引用作为从PL/SQL调用Windows的API函数的一个句柄	
Arrow Style	线	确定线是否带箭头。可取值为：None(缺省值)、Start、End、Both ends、Middle to Start、Middle to End	
Audio Channels	声音项	在数据库中存储的声音项的通道数量。可取值为：Automatic、Mono、Stereo。内部过程Write_Sound_File也使用这个属性，把声音数据按照指定的通道类型写入文件系统的一个文件中	

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
Automatic Column Width	LOV	确定是否自动地设置LOV列的宽度。Yes表示设置列宽度为Display Width属性的最大值，或者是Column Title属性中文本的长度。No(缺省值)表示设置列宽度为显示宽度	
Automatic Display	LOV	确定当用户导航到带有LOV的项时是否立刻显示LOV	
Automatic Position	LOV	确定是否在带有LOV项附近显示LOV	
Automatic Query	关系	当Deferred属性为Yes(缺省值)时，Automatic Query属性设置为Yes，表示当用户导航到从属块时，运行状态的表单自动地执行查询。No表示用户必须显式地启动Execute Query函数	
Automatic Refresh	LOV	确定是否用户在每次调用LOV时执行位于LOV中的查询(Yes是缺省值)。No值允许用内部子程序对记录组进行操作，而不是自动地从数据库中查询LOV的列，或者为避免太频繁地刷新LOV，因为大多数LOV使用记录组。通常最好把这个属性设置为No，除非数据库中的数据十分经常地改变。指示记录组有记录的内部标志在记录组第一次使用时为真，如果这个内部标志被设置，或者如果本属性被设置为Yes的LOV使用并且关闭记录组(清除组中的记录)，则LOV一直进行查询。对可以在数据库中进行修改的数据，通常保持这个属性设置为Yes，以避免在这个属性设置为No时发生的联结和交互作用。如果把它设置为No，要认真地试验改变LOV使用的顺序，以确保每件事情预期正常进行	GS
Automatic Select	LOV	确定当可选择的记录减少到单个记录时是否自动地解除LOV(Yes),或者等待用户来解除LOV(No,缺省值)	
Automatic Skip	文本项、LOV	确定是否在输入项的最后一个字符后把游标移到下一项(缺省值是No)。对带有LOV的项，确定是否在单击LOV中的记录后移动游标	GS
Background Color	项、制表页、画布、窗口、单选按钮	对象的背景的颜色	GS
Bevel	文本、图表、图像或自定义项，画布	对象边界的斜面风格：LOWERED(缺省值), RAISED,INSET, OUTSET,NONE。如果画布或项有滚动条，应该把边界的风格设为 RAISED或LOWERED，则效果比较好一些	GS
Block Description	数据块	显示在数据块菜单中的文本。如果使用块菜单的Listed项，能够在菜单中相识这个数据块。缺省为数据块的初始名字	
Bottom Title	编辑器	出现在编辑窗口的按钮上的标题 (最多可有72个字符)	
Bounding Box Scalable	图形文本	确定在按比例调整文本时，是否要按比例调整文本对象的边框(文本周围的不可见的框)，缺省值为Yes	

China-pub.com
下载

(续)

属 性	对 象	描 述	Get/Set
Button 1 Label	警告框	警告中显示的三个按钮中的第一个按钮的文本标签，缺省为OK。如果把这个标签设为NULL，则警告不再显示这个按钮	S
Button 2 Label	警告框	警告显示的三个按钮中的第二个按钮的文本标签，缺省为Cancel。如果把这个标签设为NULL，则警告不再显示这个按钮	S
Button 3 Label	警告框	警告显示的三个按钮中的第三个按钮的文本标签，缺省为NULL。如果这个标签为NULL，则警告不再显示这个按钮	S
Calculation Mode	项	说明该项是否是一个计算项，如果是，计算项是哪种类型：None(缺省值)表示该项不进行计算，Formula表示该项的值按照Formula属性中的公式进行计算，Summary表示该项按照Summary Function属性中的求和操作进行计算。该函数使用来自于Summary Data Block和Summary Item属性中说明的一个数据块项的数据计算一个值	
Canvas	项	显示项的画布。NULL表示该项没有画布。对不属于一个具体画布的项，Forms Runtime不会在任何地方显示该项，可以认为该项被显示在一个空的画布上。Layout Wizard在向导的画布屏幕中设置这个属性。如果用工具调色板建立该项，则Layout Editor对它进行设置。但是如果用Add按钮建立该项，则该值缺省为NULL	
Canvas Type	画布	画布的类型：Content(缺省值), Stacked, Vertical Toolbar Canvas, Horizontal Toolbar Canvas	
Cap Style	图形	图形中边界相交的风格：Butt(缺省值), Round, Projecting	
Case Insensitive Query	文本项	确定Forms Runtime是否在查询过程中忽略这种项的书写形式(缺省值No)。Oracle Developer优化查询，允许使用索引，而不喜欢通常带有大写转换查询的形式	GS
Case Restriction	文本项	输入到文本项的文本的形式：MIXED(缺省值)、UPPER、LOWER。例如，使用这个属性可以保证所有的文本以大写形式写入数据库或以大写形式显示	GS
Character Cell WD/HT	表单	说明使用Coordinate System时字符单元的宽度和高度。宽度和高度单元在Real Unit属性中，通过Coordinate System对话框说明	
Chart Subtype	图表项	图表类型的一个变种：Column(缺省值，应用于列的图表类型)	
Chart Type	图表项	基本的图表类型：Column(缺省值)、Pie、Bar、Table、Line、Scatter、Mixed、High-low、Double-y、Gantt	
Check Box Mapping of Other Values	复选框项	对于来自于数据库的值的解释。该值既不是Value When Checked属性中的值，也不是Value When Unchecked属性中的值，可取值有：Not Allowed(缺省值)、Checked或Unchecked	

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
Clip Height	图像	图像对象从底部裁减后在 Coordinate System 单元的高度。缺省值是图像对象的原始高度值。如果尺寸比缺省高度值小，表单按照新高度从底部剪裁图像	
Clip Width	图像	图像对象从右边裁减后在Coordinate System单元的高度。缺省值是图像对象的原始宽度值。如果尺寸比缺省宽度值小，表单按照新宽度从右边剪裁图像	
Clip X Position	图像	在Coordinate System单元中，从图像左边界进行剪裁的大小，缺省值为0	
Clip Y Position	图像	在Coordinate System单元中，从图像顶部边界进行剪裁的大小，缺省值为0	
Close Allowed	窗口	允许(缺省值)或禁止窗口系统菜单的 Close命令，通过窗口菜单或图标化关闭框。还需要编写一个When-Windows-Closed触发器，该触发器调用Hide_Window或Exit_Form来真正地关闭窗口。在微软的Windows中，MDI主窗口自动调用Exit_Form方法	
Closed	弧形图形	确定一个弧形图形是否关闭	
Column Mapping properties	LOV	单击More显示Column Mapping对话框，允许输入该列对应LOV的信息：Column Name(关联到记录组中的名字， LOV从该记录组得到它的数据)、Column Title(LOV显示在列上方的字符串)、Display Width(在Coordinate System单元中列的宽度)、Return Item(当用户选择一个值并单击OK时，LOV分配给该列的值的项的名字)。返回项可以是数据块项(block.item)。参数(Parameter.name，前面没有冒号)，或公共变量(Global.name，前面没有冒号)	
Column Name	除控制按钮、图表、VBX控件、ActiveX控件以外的项	连接该项与数据块表中的列。如果该列是一个对象类型列或是一个REF列，该列名字使用圆点记法来限定名字，例如Address.Zip表示在Person表中的Address对象中的Zip列	G
Column Specifications	记录组	在More按钮上单击，将显示Column Specification对话框。Column Name列表显示记录中列名的集合。Column Type显示当前被选择列的类型(缺省为Char、Number、Date、Long)和长度(缺省为数据库的宽度或对静态组至多为 30)。Column Values列表显示被选择的记录值，或静态记录组类型	
Comments	所有对象	可以用于任何目的的文本，它纯粹是内部文本，Forms Runtime从不显示它，但是如果产生表单文档，则把它打印出来。通常把有关该对象的信息放在此处，以后用来维护该对象	
Communication Mode	报表、图表项	通讯方式：异步或同步。异步方式立即将控制返回给表单，而报表或图形作为一个独立的线程或过程运行。同步方式将运行报表或图形，不返回控制权，直到报表或图形程序完成。对于图表项，如果表单中正在更新该图表项，必须是同步方式	

China-pub.com
下载

(续)

属 性	对 象	描 述	Get/Set
Compress	声音项	确定在把声音对象读入表单中时是否对该对象进行压缩。缺省是在声音数据文件中使用压缩设置，如果有的话	
Compression Quality	图像项	在把图形读入图形项中时使用的压缩级别。可取值为：None(缺省值)、Minimum、Low、Medium、High、Maximum	GS
Conceal Data	文本项	隐藏用户输入到该项的字符。对口令和类似用途，使用本属性。缺省值是No	GS
Console Window	表单	在窗口的底部显示控制或状态条的窗口的名字。在Windows上，MDI应用窗口一直显示控制条，但是为了显示控制状态条，仍然必须在表单对象上设置这个属性，即使它在指定的窗口中不会真正地进行显示。将这个属性设置为NULL禁止控制条。缺省值为WINDOW1	
Control Help	ActiveX控制项	在More上单击显示OCX的帮助文件，如果有的话	
Control Properties	ActiveX控制项	在More上单击显示该控制的属性工作表	
Coordinate System	表单	在More上单击，显示Coordinate System对话框。该对话框允许说明坐标的类型 (字符或缺省为实数)，实数单位(缺省为点，还有像素、英寸、厘米)，以及字符宽度和高度 (缺省值依赖于单位)。只有在字符方式的应用程序中，或者是要求在字符和GUI系统之间是完全可移植的情况下，才使用字符类型	
Copy Value from Item	除控制按钮、图表和图像以外的项	Forms Runtime在建立数据项时赋予该数据项的值的数据源。使用<数据块>.<数据项>格式。在从属块的项上，映射到主块的外部关键字有这个属性，被说明为主块项的名字。然后关系查询使用主块的值去查询从属块的记录行	G
Current Record Visual Attribute Group	表单、 数据块、项	当一个数据项是当前被显示数据块记录的一部分时用来格式化该项的可视属性。在数据块或表单上设置该属性对于数据块或表单中所有的项起作用。应该避免在表单层说明这个属性，因为它同样影响工具条和所有其他的表单元素	GS
Cursor Mode	表单	无用属性，不要使用这个属性。Open(缺省值)在事务处理过程中使游标打开。 Close在提交或回滚时关闭游标	GS
Custom Spacing	文本(图形)	文本在画布上的间距，使用Coordinate System单元，缺省值为0	
Dash Style	图形	图形对象的边缘风格，可取值为： Solid(缺省值)、Dotted、Dashed、Dash Dot、Double Dot、Long Dash、Dash Double Dot	
Data Block Description	数据块	描述数据块的文本	
Data Query	层次树	树的基本查询数据源，缺省值为NULL	
Data Source Data Block	图表项、报表	报表或图形显示的数据源，在报表或图形显示模块中替换任何的内部数据模板。如果是空值，则通知Forms Runtime从被调用模块取得自己所拥有的数据进行显示	

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
Data Source X Axis	图表项	数据块的列，其值作为图形显示的X坐标值	
Data Source Y Axis	图表项	数据块的列，其值作为图形显示的Y坐标值	
Data Type	记录组、参数、各种类型项	Form Builder的项或参数或记录组中的列的数据类型。为复选框、显示项、列表项、单选按钮组，文本项和用户自定义项说明一种类型。只能使用CHAR、DATE、DATETIME和NUMBER等类型(以及BLOB、CLOB、NCLOB和BFILE blob等类型)。如果列是一个数据库的列，则类型必须是相容的。CHAR相当于VARCHAR2，但只支持2000个字符。DATE是任何的有效的日期数据。DATETIME是任何有效的日期和时间数据。NUMBER对应于数据库的NUMBER类型。该类型是浮点数，不能把逗号放入数据中	
Database Block	数据块	确定数据块是否从数据库中得到数据 (类型取决于DML的Data Target Type属性)	
Default Alert Button	警告框	决定哪一个告警按钮是缺省的按钮，这个按钮看上去有别于其他按钮，当用户进行“default”操作时激活该按钮，而不管操作平台是否定义为选择方式。Button 1是缺省按钮，而Button 2或Button 3是可选择的	
Default Button	控制按钮项	确定这个按钮项是否是缺省按钮。这个按钮看上去有别于其他按钮，当用户进行“default”操作时激活该按钮，而不管操作平台是否定义为选择方式。缺省值是No	
Default Font Scaling	表单	确定表单字体是否使用对应字符的显示比例。缺省值是Yes	
Defer Required Enforcement	表单	确定是否推迟强制执行项的Required属性，直到进行记录有效性检查为止，允许用户导航到其他项上，但不能离开该记录或该块。缺省值是No，但是把该值设置为Yes能够增强应用程序的可用性	GS
Deferred	关系	确定在一个因协调原因的事件发生时是否读取从属记录。这个属性与Automatic相互作用来确定协调活动	
Delete Allowed	数据块	确定用户是否可以从数据块删除记录	GS
Delete Procedure Arguments	数据块	当DML的Data Target Type属性被设为Procedure时，这个属性包含有参数的名字、类型和值，这些参数传递给在Delete Procedure Name属性中命名的当前过程，该过程用来删除数据。Forms Runtime调用这个过程执行Delete Record操作，而不是生成一条SQL的DELETE语句。缺省值为NULL	
Delete Procedure Name	数据块	当DML的Data Target Type属性被设为Procedure时，这个属性包含有执行删除一个记录操作的当前过程的名字。Forms Runtime调用这个过程执行删除记录操作，而不是生成一条SQL的DELETE语句。缺省值为NULL。参见第6章中程序的封装的有关细节	

China-pub.com
下载

(续)

属 性	对 象	描 述	Get/Set
Delete Procedure Result Set Columns	数据块	当DML的Data Target Type属性被设为Procedure时，单击在这个属性上的More，显示Delete Procedure Result Set Columns对话框，这个对话框包含有一些列的名字，Forms Runtime把现有过程的结果集复制到这些列上。如果删除现有过程的返回值，比如删除数值或一个结果代码，应该说明这些属性	
Delete Record Behavior	关系	Forms Runtime如何把在主记录中的删除操作传送给它的从属记录，可取值有：Non-Isolated(缺省值)、Isolated或Cascading。Non-Isolated表示当从属记录存在时用户不能删除一个主记录。Isolated表示用户可以删除一个主记录而不会影响从属记录。Cascading表示如果用户删除一个主记录，则Forms Runtime删除该主记录的从属记录	GS
Detail Block	关系	从属块的名字	G
Detail Reference Item	关系	当关联的类型是REF时，在从属块中连接主数据块的REF项	GS
Direction	各种对象	说明对象布局的方向。可取值为：缺省、Right-to-Left、Left-to-Right、Applies to Form、Alert、Data Block、LOV、Window、Canvas、Check Box、Button、Radio Group、List Item	GS
Display Hint Automatically	除图表、显示和自定义项以外的项	确定当输入焦点进入该项时是否自动地显示Hint属性中的文本内容，缺省值为Yes。如果将此属性设置为No，则Forms Runtime只是在输入焦点进入该项并且用户选择Help时才显示提示。如果Hint属性为NULL，则这个属性不起作用	GS
Display in‘Keyboard Help’	触发器	确定在Forms Runtime的键帮助屏幕上是否显示Keyboard Help Description域中的文本，缺省值为No	
Display Quality	图像项	Forms Runtime用于显示图形质量的级别，可取值为：High(缺省值)、Medium、Low。如果出现资源限制问题，比如在Windows系统上只有少量的GDI资源，可以通过将这个属性设置为Medium或Low的方法，减少资源的消耗	
Distance Between Records	项	多记录数据块(显示的记录数大于1)中各项之间距离的间隔数，以Coordinate System单位为单位。缺省值为0	
Dither	图像(图形)	确定在一个样板图形中是否抖动(颜色平滑渐变)	
DML Array Size	数据块	Forms Runtime在对数据库进行插入、删除或修改操作过程中发送数据所使用数组的最大尺寸，缺省值为1。该数组的尺寸越大，则网络传输量越小，使用内存越多。如果有大量内存空间，并且进行大量的多记录事务处理，可以增加这个属性的值达到一个用户在单个事务处理中修改记录的期望值。如果Insert Allowed属性是Yes，并且设置数组尺寸大于1，则必须使用	

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
DML Array Size	数据块	Enforce Primary Key和 Primary Key属性说明一个主关键字，因为Forms Runtime在对数组数据进行操作时不能得到一个行标识。另外，如果数组尺寸设置为大于1，则Update Changed Columns Only属性要始终是No，因为该数组必须含有每个记录的所有列。如果在记录中有大的对象或图像数据，数组的大小可能会阻止 Forms Runtime使用该数组	
DML Data Target Name	数据块	想要插入、删除和修改的表的名字，缺省为NULL，并且只有在 DML Data Target Type属性设置为Table时，该值才是有效的。当从数据块中的一个视图或过程进行查询时，使用这个属性对表进行操作。在运行时刻还可以使用 Get和Set内部子程序修改这个表的名字(可是需要相当小心这种类型的程序设计)	GS
DML Data Target Type	数据块	从数据块进行插入、修改和删除操作的目标的类型：Table(缺省)、Procedure或Transactional Trigger。可以在DML Data Target Name属性中设置表的名字。如果说明为 Procedure，则必须在插入、修改和删除过程属性中提供过程名和参数	G
Edge Background Color	图形	图形对象边界背景颜色，缺省为 NULL	
Edge Foreground Color	图形	图形对象边界前景颜色，缺省为 NULL	
Edge Pattern	图形	图形对象边界的样式，缺省值为 NULL	
Editor	文本项	编辑文本时使用的编辑器，缺省值为 NULL，表示使用Forms Runtime的缺省编辑器。如果为SYSTEM_EDITOR，Forms Runtime查找表单FORMS60_EDITOR注册变量，并使用在那说明的程序来编辑文本。否则可以指定表单模块中已经定义的一个编辑器对象的名字	G
Editor X Position	文本项	在Editor属性中说明的文本编辑器的左上角的水平方向(即x轴)的坐标，忽略Editor对象的Position属性的值。缺省值为0，表示使用Editor Position属性	G
Editor Y Position	文本项	在Editor属性中说明的文本编辑器的左上角的垂直方向(即y轴)的坐标，忽略Editor对象的Position属性的值。缺省值为0，表示使用Editor Position属性	G
Elements in List	列表项	在More上单击显示Elements in List的对话框，该对话框用来输入每个列表框元素的标签和值。如果List Item Value是空，则对应的值为NULL	
Enabled	标签页	确定是否正常显示标签页 (Yes,缺省值)或使它为灰色(No)	GS
Enabled(Item)	除控制按钮、图表和显示项以外的项	确定用户是否可以用鼠标操作这个项，缺省值为Yes。把这个属性设置为No，则该项变灰。如果希望该项不变灰但又不能被改变，则把 Insert Allowed和Update Allowed属性设置为No。设置Enabled为No同样也把Keyboard Navigable属性设为No	GS

China-pub.com
下载

(续)

属 性	对 象	描 述	Get/Set
End Angle	弧形图形	弧形图形的结束角度，使用水平坐标轴作为起点，缺省值为180	
Enforce Column Security	数据块	确定Forms Runtime在启动表单时是否查看用户在数据库中修改列的权利，以确定是否允许该用户对数据块中的对应项通过关闭Update Allowed项属性的方法进行修改。缺省值为No	G
Enforce Primary Key	数据块	确定是否在数据库中插入行或修改行操作之前检查指定的主关键字的值，缺省值为No。当基础表没有进行PRIMARY KEY约束关系定义而又要坚持这种约束关系时，或者当把DML Array Size属性设置为大于1并允许进行插入时，使用这个属性。如果设置这个属性为Yes的话，那么必须在数据块中至少将一个项的Primary Key属性设置为Yes	GS
Enterable	数据块	确定用户是否能够操纵在该块中的一个项。这个属性缺省值为Yes，但是如果在该块中没有可输入的项(Keyboard Navigable设置为Yes的项)，则自动被设置为NO	
Execution Hierarchy	触发器	确定在对象层次结构中的一个较高级别上存在另外的同名的触发器时，当前触发器执行的时机，可取值为：Override(缺省值)、Before、After。Override仅执行当前触发器。Before先执行当前触发器，然后在执行较高级别上的触发器。After先执行较高级别上的触发器，然后执行当前触发器。Oracle只对大量的管理和测试推荐使用这个功能，例如它可以开发出难以维护的应用系统	
Execution Mode	图表项或报表	Forms Runtime执行报表或图形显示的方法，可取值为：Batch(缺省值)或Runtime。用Batch方式执行报表或图形时不与用户进行交互，而Runtime允许与用户进行交互	
Filename	表单、图表项	表单或图形显示模块的文件名。不能设置这个属性。Form Builder在从文件读取表单模块时设置它。它对应于应用程序属性Current_Form	G
Fill	弧形图形	Arc对象的填充形状，可取值为：Pie(缺省值)或Chord。Pie型填充从圆的中心点开始进行。Chord型填充从由一个终点画到另一个终点的线段开始进行	
Fill Pattern	项、制表页、画布、窗口	用于填充该对象的模式(缺省值为无模式)	GS
Filter Before Display	LOV	确定在显示LOV之前是否显示一个Query Criteria对话框(缺省值为No)。用户可以将数值输入到对话框中以进一步限定查询，LOV将该标准添加到记录组的SQL语句中	
Fire in Enter-Query Mode	触发器	确定是否在Enter Query方式中执行该触发器，缺省值为No。这个属性只对确定的触发器起作用：Key-<功能键>、On-Error、On-Message和除以下触发器以外的When-<event>触发器：	

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
Fire in Enter-Query Mode	触发器	When-Database-Record, When-Image-Activated, When-New-Block-Instance, When-New-Form-Instance, When-Create-Record, When-Remove-Record, When-Validate-Record和 When-Validate-Item	
First Navigation Data Block	表单	在Forms Runtime打开该表单时操纵的块的名字，缺省为在Object Navigator中表单里的第一个块。既可以按照所喜欢的顺序安排块，也可以设置这个属性，但通常在When-New-form-Instance触发器中有计划地设置这个属性以改变表单模块的行为特征	GS
Fixed Bounding Box	文本(图形)	确定文本的边框是否保持不变，缺省值为No(根据输入的文本进行扩展)。如果这个属性是Yes，则Width和Height属性确定边框的大小	
Fixed Length	文本项	强制用户输入Maximum Length属性说明的字符数，把该区域设计成一个特定的、固定长度的区域。缺省值为No	GS
Font Name	项、制表页、画布、窗口	在对象中用于文本的字体集	GS
Font Size	项、制表页、画布、窗口	在对象中用于文本的字体尺寸	GS
Font Spacing	表单	如果坐标系被设置为Character，这个属性说明表单在运行时把显示设备的对应的字符比例作为缺省值	
Font Spacing	项、制表页、画布、窗口	字体的宽度。缺省值是Unspecified，表示使用缺省的字体设置	GS
Font Style	项、制表页、画布、窗口	字体的风格，可取值为：Plain、Italic、Oblique、Underline、Outline、Shadow、Inverted、Overstrike或Blink。缺省值是<Unspecified>，表示使用缺省的字体设置	GS
Font Weight	项、制表页、画布、窗口	字体宽度。可取值为：Ultralight、Extralight、Light、Demilight、Medium、Demibold、Bold、Extrabold、Ultrabold。缺省值是<Unspecified>，表示使用缺省的字体设置	GS
Foreground Color	项、制表页、画布、窗口	对象的前景颜色，如果对象是项，则指该项显示的文本的颜色	GS
Form Horizontal Toolbar Canvas	表单	在MDI应用程序窗口上作为一个水平工具条显示的画布(只适用于Microsoft Windows)。该画布必须已经将Canvas Type设置为Horizontal Toolbar	
Form Vertical Toolbar Canvas	表单	在MDI应用程序窗口上作为一个垂直工具条显示的画布(只适用于Microsoft Windows)。该画布必须已经将Canvas Type设置为Vertical Toolbar	
Fromat Mask	文本项	该项显示文本的格式，也是该项作为输入文本所要求的格式。参看稍后的“格式屏蔽 (表单Masks)”部分，了解格式屏蔽符号的引用方法	GS

China-pub.com
下载

(续)

属 性	对 象	描 述	Get/Set
formula	项	一个PL/SQL表达式(不是一条语句-不要输入分号或者建立一条赋值语句)，当Calculation Mode属性是Formula时，Forms Runtime用于对该项进行计算。没有缺省的表达式	
Frame Alignment	帧	如何在帧的宽度范围内排列对象，可取值为：Fill(缺省值)、Start、End、Center、Column。只在Layout Style是表单时有效	
Frame Title	帧	在帧的边界上显示的文本，缺省值为 NULL	
Frame Title Alignment	帧	Frame Title属性中的文本内容的排列方式，可取值为：Start(缺省值)、End、Center。与画布的Direction属性有关	
Frame Title Font Background Color	帧	显示Frame Title属性中的文本的背景颜色。缺省为标准的操作系统的颜色，通常是黑色	
Frame Title Font Foreground Color	帧	显示Frame Title属性中的文本的颜色。缺省为标准的操作系统的颜色，通常是黑色	
Frame Title Font Name	帧	显示Frame Title属性中的文本的字体的名字。缺省为标准的操作系统字体	
Frame Title Font Size	帧	显示Frame Title属性中的文本的字体的大小。缺省为标准的操作系统大小	
Frame Title Font Spacing	帧	显示Frame Title属性中的文本的字体的字符间距。缺省为标准的操作系统字体间距	
Frame Title Font Style	帧	显示Frame Title属性中的文本的字体的字形(例如斜体)。缺省为标准的操作系统字体字形	
Frame Title Font Weight	帧	显示Frame Title属性中的文本的字体的粗细(例如粗体)。缺省为标准的操作系统字体的粗细	
Frame Title Offset	帧	从帧的边角到帧标题(参见Frame Title属性)的距离，缺省值是在Coordinate System单位中的2个字符	
Frame Title Reading Order	帧	帧标题(参见Frame Title属性)的显示方向，可取值：Default、Left-to-Right、Right-to-Left	
Frame Title Spacing	帧	帧标题的每个边所保留的间隔位置的数量，缺省值为1个字符，以Coordinate System单位为单位	
Frame Title Visual Attribute Group	帧	命名的可视属性，从其中能获得对帧标题的各种字体、颜色和填充的设置。参见第 10章。	
Graphics Type	图形	图形对象的类型，可取值为： Arc、Chart、Group、Image、Line、Polygon、Rectangle、Rounded Rectangle、Symbol、Text	
Height	画布、项、编辑器、LOV、窗口	对象的高度，以Coordinate System单位为准。缺省值随对象类型而变化	GS
Hide on Exit	窗口	对非模式的(与模式的截然相反)窗口，确定在用户导航到另一个窗口中的项上时，是否隐藏该窗口，缺省值为No	GS
Highest Allowed Value	文本项	用户可以输入的最大值(按字符、数值或日期/时间顺序)，缺省值为NULL	G
Hint	项	当输入焦点在该项上时， Forms Runtime显示在控制台或状态栏上的文本。如果该项是使用Data Block Wizard创建的，则缺省值是：”Enter value for:<项> ”,如果没有提示，则缺省值为NULL	G

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
Horizontal Justification	图形文本	文本对象的调整。可取值为：Start(缺省值)、End、Left、Right、Center	
Horizontal Margin	帧	帧和帧内对象的左、右边界之间的距离（页边距)。缺省值是1个字符，以坐标系单位为单位	
Horizontal Object Offset	帧	对象之间的水平距离，缺省值为2个字符，以Coordinate System单位为单位	
Horizontal Origin	图形文本	文本对象相对于原点的水平位置。可取值为：Left(缺省值)、Right、Center	
Horizontal Toolbar Canvas	窗口	在窗口上作为一个水平工具条显示的画布。该画布必须把它的Canvas Type属性已设置为Horizontal Toolbar，并且必须把它的Window属性已经设置为这个窗口。缺省值为NULL	
Icon Filename	控制按钮项、窗口	包含要显示的位图图标的图标文件的名字。不要加上扩展名，使用UI60_ICON注册变量来说明所有图标文件的路径	GS
Iconic	控制按钮项	使得按钮成为肖象按钮，即用图像代替按钮上的文本的一种按钮，缺省值为No。如果这个属性设置为Yes，由属性Icon Filename确定所要显示的肖象	G
Image Depth	图像项	要显示的图形的样式，可取值为：Original(缺省值)、Monochrome、Gray、LUT(查找表)、RGB	GS
Image Format	图像项	在数据库中存储图像的格式，可取值为：TIFF(缺省值)、BMP、CALS、GIF、JFIF、PICT、RAS、TPIC。这个这种格式代替该项的初始格式	G
Implementation Class	Bean Area、复选框、列表框、控制按钮、单选组按钮、文本项	对Bean Area是一个JavaBean的类名，对其他任何项是一个自定义的执行过程。没有缺省值	
Include REF Item	数据块	对一个主数据块，在这个块中包括一个特殊的、隐含的、称为REF的项，该项用来协调建立在REF连接上的一个主从关系。缺省值为No	
Inherit Menu	窗口	确定该窗口是否应该显示表单的菜单。适用于支持窗口菜单的平台。缺省值为：Yes	
Initial Keyboard State	显示、文本项	设置键盘状态，用户能够立即地开始键入，而不用转换键盘状态。可取值为：Default(参看Reading Order属性)、Local(从右到左)、Roman(从左到右)	
Initial Menu	表单	在菜单模块中用来作为该表单菜单的菜单名字，缺省值为空白。在这里指定一个名字，允许越过在Main Menu属性的菜单模块中的缺省的菜单，使用那个模块另外一个菜单，给出对该表单的不同的子模式显示不同菜单的方法	
Initial Value	复选框、显示、列表、单选组、文本和用户区项	当创建一个新的记录时，分配给该项的缺省值，缺省为NULL。可以给出一个值(数值型、字符串型、日期型)、一个表单项(:<块名>.<项名>)、一个全局变量(:GLOBAL.<变量名>)、一个参数(:PARAMETER.<参数名>)，或者一个序列(SEQUENCE.<名字>.NEXTVAL)	

China-pub.com
下载

(续)

属 性	对 象	描 述	Get/Set
Insert Allowed(Block)	数据块、项	确定用户是否可以在一个数据块中插入记录或者在一个新记录中修改该项，缺省值为 Yes。对于一个项，把这个属性设置为 No，可以防止用户对该域进行输入或修改，而不用关闭那个域	GS
Insert Procedure Arguments	数据块	当DML Data Target Type属性被设置为Procedure时，这个属性包含参数的名字、数据类型和值，这些参数要传送给在 Insert Procedure Name 属性中命名的用于插入数据的已存储的过程。Forms Runtime调用这个过程来进行插入记录操作，取代生成一条SQL INSERT语句。缺省值是NULL	
Insert Procedure Name	数据块	当DML Data Target Type属性被设置为Procedure时，这个属性包含执行插入一个记录的已存储的过程的名字。Forms Runtime 调用这个过程，代替生成一个SQL INSERT语句。缺省值是NULL。参看第6章中的“程序封装”段的详细内容	
Insert Procedure Result Set Columns	数据块	当DML Data Target Type属性被设置为Procedure时，单击该属性上的“More”，来显示Insert Procedure Result Set Columns对话框，这个对话框中包含有Forms Runtime对已存储的过程的结果集进行复制的列的名字。如果插入操作的已存储的过程有返回值，例如一个新的值或者是一个结果代码，应该说明这个属性	
Interaction Mode	表单	确定在查询过程中用户如何与表单进行交互，可取值有：Blocking(缺省值)或Non-Blocking。Blocking防止用户调整大小或者对表单做其他任何事情，直到查询结果被取出。Non-Blocking容许用户在读取过程中与表单进行交互。它对返回记录要花费很长时间的查询是非常有用的	G
Isolation Mode	表单	如何实施一个事务处理，可取值有：Read Committed(缺省值)或Serializable。几乎始终允许这个属性设置为 Read Committed。如果把它设置为Serializable，Forms Runtime把数据库对话程序设置为隔离状态，所以该事务处理与其他所有的事务处理肯定是以串行方式进行的。实际上，这意味着如果一个用户查询和更新一个记录行却没有进行提交，而另一个用户出现，对该记录行进行了更新，并在第一个用户之前进行了提交操作，当第一个用户提交时，他(她)会得到Oracle的错误信息ORA-08177：Cannot serialize access (不能串行访问)。如果想要使用 Serializable，应该设置加锁方式进行延时。只有在几乎没有覆盖更新可能的情况下，才可以使用 Serializable	
Item Type	项	项的类型，可取值有：Text Item(缺省值)、ActiveX Control、Bean Area、Chart Item、Check Box、Display Item、Hierarchical Tree、Image、List Item、OLE Container、Push Button、Radio Group、Sound、Text Item、User Area、VBX Control	

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
Join Condition	关系	定义在主数据块和从数据块之间进行连接的表达式	
Join Style	图形	在图形中把两条线连接到一起的方法，可取值有：Mitre(缺省值)、Bevel、Round	
Justification	显示和文本项	在项中文本对齐的方法，可取值有：Start(缺省值)、End、Left、Center、Right	GS
Keep Cursor Position	文本项	当用户离开该文本项时保存游标的位置，当用户返回到该项时恢复游标位置。缺省值是 No	GS
Key Mode	数据块	Forms Runtime如何对非Oracle数据源的数据库中的记录行进行标识（关键字），可取值有：Automatic(缺省值)、Non-Updateable、Unique、Updateable。Oracle使用ROWID来标识记录行，其他数据库使用关键字值。Automatic表示如果ROWID可用则使用ROWID，如果没有ROWID则使用主关键字。Non-Updateable关键字模式在一个UPDATE SET子句中不包含主关键字列，所以不能更新数据库中的主关键字列。Unique一直使用ROWID。Updateable允许用户更新主关键字的值。如果正在使用 Non-Updateable或Updateable，则需要把Enforce Primary Key设置为Yes，并且必须至少指定一个项作为主关键字的一部分，通过把该项的 Primary Key属性设置为Yes的方法来实现。当在使用非 Oracle数据库时使用这个属性	GS
‘Keyboard Help’Text	触发器	当把Keyboard Help中的Display项设置为Yes时出现在键帮助屏幕上的文本。缺省值为空白，表示显示该键的缺省描述	
Keyboard Navigable	除图表和显示项以外的项	确定用户或者Forms Runtime在导航过程中是否能够把焦点移动到该项上，缺省值为 Yes。如果把该属性设置为No，Forms Runtime跳过该项并且进入到导航序列中的下一个可导航的项。如果把Enabled属性设置为 No，则Form Builder把Keyboard Navigable属性也设置为No。然而，逆过程是不成立的，把Enabled属性设置为Yes，并不能把Keyboard Navigable属性设置为Yes，所以不得不显式地改变它，要么将不能导航到该项上。把Keyboard Navigable属性设置为No可以使用Go_Item来导航到一个项上。如果在实例级使用内部子程序设置 Keyboard Navigable属性，对它无效，除非项目级的属性被设置为同样的值	GS
Keyboard State	项	设置所支持的各国的国别键盘状态，可取值有：Any(缺省值)、Roman Only或Local Only。用户可以使用一个键在键盘状态之间进行切换	
Label	控制按钮、复选框和单选按钮项、标签页	一个按钮、复选框或单选按钮上显示的文本标识，或者是一个标签页在它的标签上显示的文本标识	GS
Last Query	数据块	Forms Runtime对该数据块的最后的查询发出的SQL语句	G

China-pub.com
下载

(续)

属 性	对 象	描 述	Get/Set
Layout Data Block	帧	拥有帧重排的项的数据块，每个块只能有一个帧，而每帧只能在一个块中。缺省值为NULL	
Layout Style	帧	帧排列项的方法，可取值为：Form(缺省值)或Tabular。在Form中，帧把项重新排列成两列，提示放在每个项的左边。在Tabular中，帧把项彼此相连的排列在一行上，提示放在每个项的上方	
Line Spacing	图形文本	在文本对象中的线的间距，可取值为：Single(缺省值)、One-and-a-half、Double、Custom(参看Custom Spacing属性)	
Line Width	图形	图形对象边界的宽度，以点为单位	
List Item Value	单选按钮项	当该组中的单选按钮被设置时的单选按钮项的值，缺省为NULL(一个NULL值)	
List Style	列表项	列表项的类型，可取值为：Poplist(缺省值)、Combo Box或Tlist。Poplist是一个下拉列表，Combo Box是一个标准的组合框下拉列表，Tlist是一个滚动的元素列表。对所有类型的列表，数据结构保持相同的形式	
List Type	LOV	确定如何引用记录组，可取值为：Record Group(缺省值)或Old(一种旧方式的LOV，对新的应用程序不再支持)。如果该值为Record Group，必须在Record Group属性中选择该记录组	
List of Values	文本	当用户按下LOV键或菜单项时为文本项显示的LOV的名字。当把一个LOV与一个项进行关联时，Forms Runtime在控制状态行上显示一个信息，指明该项有一个LOV	G
List X Position	文本项	对在List of Values属性中有LOV的项，其LOV相对于屏幕左上角的水平位置，缺省值为0。如果X和Y两个位置都是0，LOV对象按照优先次序决定坐标值	G
List Y Position	文本项	对在List of Values属性中有LOV的项，其LOV相对于屏幕左上角的水平位置，缺省值为0。如果X和Y两个位置都是0，LOV对象按照优先次序决定坐标值。(在原书中本属性与前一个属性的描述完全一样，但其中的“水平位置”似应为“垂直位置”——译者)	G
Listed in Block Menu/ Block Description	数据块	确定是否在块菜单中列出该数据块，用户可以通过按下Block Menu键来访问块菜单。该菜单容许用户使用在这个菜单中选择该块的方法，导航到数据块中的第一个项。文本来自于Block Description属性	
Lock Procedure Arguments	数据块	当DML Data Target Type属性被设置为Procedure时，这个属性包含有参数的名字、数据类型和值，这些参数要传送给在Lock Procedure Name属性中命名的存储过程，用来对一个数据行加锁。Forms Runtime对Lock Record操作调用这个过程，代替生成一个SQL SELECT FOR UPDATE语句。缺省值为NULL	

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
Lock Procedure Name	数据块	当DML Data Target Type属性被设置为Procedure时，这个属性包含有一个存储过程的名字，该过程执行对一个行的加锁操作。Forms Runtime调用这个过程，以代替生成一个 SQL SELECT FOR UPDATE语句，缺省值为NULL。	
Lock Procedure Result Set Columns	数据块	当DML Data Target Type属性被设置为Procedure时，在这个属性上单击More，显示Lock Procedure Result Set Columns对话框，其中包含一些列的名字，Forms Runtime把存储过程的结果集复制到这些列中。如果用于加锁的存储过程有返回值，例如一个结果代码，则将要说明这些内容	
Lock Record	文本项	当用户或一个触发器修改项的值时，确定是否要试图对数据库的行加锁，缺省值是 No。修改一个非数据库的项时，使用这个属性来锁定数据库的行	GS
Locking Mode	数据块	Forms Runtime在什么时间得到对行加锁的数据库，可取值有：Automatic(缺省值)、Immediate或Delayed。Automatic和Immediate对一个Oracle服务器来说有相同的作用：一旦用户改变一个文本项中的值，或是用户按下 Lock Record 键，Forms Runtime就锁定行。Delayed表示只是在真正地把记录写入到数据库的时候，Forms Runtime才锁定行。多数情况下，都应该使用Automatic缺省值。然而，如果遇到在高传输速率环境中的加锁问题，或者希望把 Isolation Mode属性设置为Serializable，应该把Locking Mode属性设置为Delayed。但是，如果另外的事务处理在此次查询和提交之间对值进行了修改，Forms Runtime将给用户一个错误信息，并且提交失败。然后用户将需要再次执行提交操作	GS
Lowest Allowed Value	文本项	对该文本在一个强制范围内的最小值。如果输入的值小于这个值，则Forms Runtime的验证进程失败。这个属性值可以是一个常量、一个表单项(:<块名>.<项名>)、一个共用变量(:GLOBAL <变量名>)或者一个参数(:PARAMETER.<参数名>)	G
Mapping of Other Values	列表框和单选按钮组项	当某件事情所赋的一个值不在该列表中或者不是所关联的单选按钮的值时，对该项设置的值，缺省值是空白，表示Forms Runtime将不允许有其他的值。注意：如果同意这个选择为空白并且一个查询返回的值不在列表或单选按钮中，Forms Runtime悄悄地拒绝这个记录，而且用户再也见不到它。如果允许通过一些其他的方法登录其他的值，可能会造成混乱	
Maximize Allowed	窗口	确定用户是否可以通过使用窗口的管理功能对窗口进行缩放或者是最大化，缺省值是 Yes	
Maximum Length	参数，除命令、图像和图表项以外的项	CHAR类型的参数或者任意类型以字符为单位的项的最大长度。对参数而言，缺省值是30；对项来说，缺省值取决于类型和其他属性的规定。	

China-pub.com
下载

(续)

属 性	对 象	描 述	Get/Set
		通常地，对数据库的项，缺省值是对应的数据库列的大小。如果类型是NUMBER，Form Builder增加2个字符，以容纳符号和小数点。LONG型项缺省值为240个字节。如果有Format Mask，或者如果存在隐含的格式掩码象日期型，Forms Runtime扩展最大长度值，以适应掩码的大小。如果使用多字节的字符集，该长度以字节为单位，而不是以字符为单位，但是以字符为单位进行截断，同时Forms Runtime显示一个警告信息	
Maximum Objects Per Line	帧	帧中的每一行上可能出现的对象的最大数目，缺省值是0，意味着没有最大值。要使这个属性是有效的，必须把Frame Style属性设置为form，把Vertical Fill属性设置为No。可以使用这个属性在帧的布局上安排较多的控制项	
Maximum Query Time	表单，数据块	如果Query All Records属性被设置为Yes，这个属性通知Forms Runtime在查询的时间超过了这个属性的值时，给出用户取消这个查询的选项	
Maximum Records Fetched	表单、数据块	如果Query Allowed属性和Query All Records属性都被设置为Yes，当被取出的记录数超过所给出的值，这个属性退出在运行状态的查询。这个属性给出了针对一个潜在的难以控制的查询的限制	
Menu Module	表单	用于这个表单的菜单模块的名字。如果把这个属性设置为Default，Forms Runtime运行内置的菜单。如果把它设置为NULL，Forms Runtime不会显示菜单。如果把Menu Source设置为Yes，这个Menu Module属性是已被编译的.MMX文件的名字。如果把Menu Source设置为No，这个属性指定在数据库中菜单模块的名字。还要查看Initial Menu属性，设置菜单模块中指定的一个菜单作为主菜单	
Menu Role	表单	安全性角色，Form Builder用于运行菜单的老式的属性。把Use Security属性设置为No，则关闭菜单的安全性检查，代替设置该角色为一个特殊的“测试”作用	
Menu Source	表单	确定是否是使用对.MMX文件直接的引用(Yes，缺省值)还是在数据库中查找菜单(No)。如果设置这个属性为Yes，则应该在Menu Module属性中提供.MMX菜单模块文件的名字。如果把它设置为No，则使用菜单模块属性指定.MMX文件的文件名	
Menu Style	表单	要显示的菜单的类型，可取值有：Pull-down(缺省值)、Full-Screen或Bar。通常应该使得这个属性设置为Pull-down。Full-Screen是一个字符模式应用程序使用的3270式样的满屏菜单。Bar是一个Lotus 1-2-3，老式的DOS类型的菜单	
Message	警告框	要显示的信息文本(无缺省值)	S
Minimize Allowed	窗口	确定用户是否可以把窗口图标化，缺省值为Yes	

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
Minimized Title	窗口	在一个图标化/最小化窗口之前出现的文本	
Modal	窗口	确定用户是否可以把焦点改变到另一个窗口，而又不释放这个窗口，缺省值No。使用这个属性，同时把Window Style属性设置为Dialog，来创建一个模式对话框	
Mouse Navigate	控制按钮、复选框、列表框和单选按钮组项	确定当用户用鼠标单击一个项时是否导航到该项上(把游标焦点移动到该项)。如果设置这个属性为No，Forms Runtime不会执行任何导航(并因此确认)，只有When-Button-Pressed或相似的触发器能进行导航。例如，可以使用这个行为去创建按钮矩阵，发出命令却不做任何的导航，与一个工具条类似	GS
Mouse Navigation Limit	表单	用户能够用鼠标导航使当前焦点离开项的程度，可取值有：表单(缺省值)、Block、Record、Item。Block表示用鼠标置能导航到当前数据块范围内的项上。Record表示用户只能进入数据块中的当前记录范围内的项上。Item意味着用户用鼠标不能导航出当前的项	
Move Allowed	窗口	确定用户是否可能把窗口移动到屏幕上的另一个位置(或者在一个触发器中的代码是否可能这样做)	
Multi-Line	文本项	确定该文本项是否允许有多行，缺省值为：No。如果设置该项有多行，必须规定它的尺寸，并要设置字体和Maximum Length属性，以显示适当数量的文本。还应该考虑Wrap Style和Show Vertical Scroll Bar属性。只能对CHAR、ALPHA，或LONG数据类型设置这个属性。如果在一个单行项中按回车键，则会导航出该项，而如果在多行的项中按回车键，则插入一个回车换行，并不会导航出该项	
Multi-Selection	层次树	确定用户是否能够一次选择多个节点。No(缺省值)意味着选择一个节点，同时取消对当前已选择的节点的选定。Yes表示用户可以选择多个节点	
Name	所有对象	对象的名字，无缺省值。名字最多可以有30个字符，必须是字母开头，可以包含字母、数字和特殊字符$、#、@、_。大小写形式没有关系。在对象的类型内部，名字是唯一的	
Navigation Style	数据块	在下一项或前一项上操作，决定当焦点分别在该数据块中的最后或第一个可导航项上时如何继续进行，可取值有： Same Record(缺省值)、Change Record、Change Block。Same Record把焦点移动到同一数据块和同一记录中的第一个可导航的项上。Change Record把焦点移动到同一数据块的下一个记录中的第一个可导航的项上(如果没有下一个记录则在一个新记录中，)。Change Block把焦点移动到下一个块和前一个块中的第一个可导航的项上	GS

China-pub.com
下载

(续)

属 性	对 象	描 述	Get/Set
Next Navigation Block	数据块	当焦点移动到下一个数据块时，Forms Runtime将导航到的数据块的名字。对于缺省情况，下一个数据块是在Object Navigator中出现的次序的下一个数据块。使用这个属性改变导航的次序，而不用重新编排数据块在Navigator中的次序。使用这个属性主要为了在运行时有计划地改变数据块的导航次序	GS
Next Navigation Item	数据块	当焦点移动到下一个项时，Forms Runtime将导航到的项的名字。对于缺省情况，下一个项是在Object Navigator中的顺序中的下一个项。有计划地或在Form Builder中如果希望重新定义项的顺序而不用重新排列数据块中的项的次序时设置这个属性	GS
Number of Items Displayed	项	多记录数据块显示的项实例(独立的项域)的数量。这个数量取代数据块的 Number of Record Display属性，在多记录数据块中允许创建单个按钮。通常不希望这个项成为一个数据块项	
Number of Records Buffered	数据块	数据块中进行一个查询的时候，Forms Runtime在内存缓存区存储记录的最小数量。缺省值是NULL，意思是Forms Runtime使用Number of Records Displayed属性的值加常数3。额外的记录放到一个临时的磁盘文件上缓冲。使用这个属性可以交替使用内存与磁盘的性能，或者通过增加这个属性值的方法，能够管理具有大量记录数的数据块，以达到提高性能的目的。还可以通过减少该属性和记录集的内存消耗量的方法，管理带有许多项(控制项、图表项等等)的数据块	
Number of Records Displayed	数据块	数据块一次显示的记录的最大数量，缺省值是1。如果把这个属性设置为一个大于1的数，则创建一个多记录的数据块	
OLE Activation Style	OLE容器	激活OLE容器项的事件，可取值有：Double Click(缺省值)、Focus-in、Manual。Double Click通过双击对象的任何地方激活该对象。Focus-in意思是导航进入OLE对象激活该对象。Manual表示用户从OLE的弹出菜单直接引用Edit或Open菜单项激活该对象，必须把 Show OLE Popup Menu属性设置为Yes，并且Object菜单显示在菜单模块中	
OLE Class	OLE容器	确定哪些OLE对象的类可能驻留在该项中，可取值有：NULL(缺省值)或已命名的类。NULL表示在运行时用户可以从注册数据库插入任何类型的OLE对象。指定一个限制用户使用的类，而不是容许任何类型的OLE类	
OLE In-place Activation	OLE容器	确定是否使得In-place Activation可用，缺省值为No。把该属性设置为Yes，能够编辑已嵌入的OLE对象；设置为No使用对象服务器强制进行外部激活	

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
OLE Inside-Out Support	OLE容器	如果OLE In-place Activation被设置为Yes，这个属性接通OLE服务器中的inside-out对象支持，允许多于一个的嵌入式对象在该OLE容器项内部有一个正在编辑的窗口	
OLE Popup Menu Items	OLE容器	从一个列表中选择要显示哪个OLE弹出菜单命令。必须设置Display和Enable选项，其中Display把命令放到菜单中，Enable允许用户选择它	GS
OLE Resize Style	OLE容器	Forms Runtime如何显示在该项中的OLE对象，可取值有：Clip(缺省值)、Scale、Initial和Dynamic。Clip裁减OLE对象以适应该容器项。Scale按比例调整对象以适合该容器。Initial只是在用户创建对象时调整对象的大小。Dynamic每当对象尺寸变化时调整它的大小	
OLE Tenant Aspect	OLE容器	确定一个OLE对象如何在该容器中出现，可取值有：Content(缺省值)、ICO或Thumbnail。Content显示对象的内容。ICO显示一个图标代表该对象。Thumbnail显示一个“极小”或缩小视图的对象。在查询已保存到该容器中的对象时的OLE对象被保存时，或当对象被自动地锁定时，该属性必须具有同样的已存在的值	
OLE Tenant Types	OLE容器	用户已经认可的设置到该容器的OLE对象的类型，可取值有：Any(缺省值)、None、Static、Embedded或Linked。Any设置表示任何的对象都可以占据该容器。None表示没有对象可以占据该容器。Static表示只有已链接的OLE对象的快照图像可以占据该容器(该链接被断开，用户不能修改该对象)。Embedded表示只有嵌入式OLE对象可以占据该容器。Linked表示只有一个链接的OLE对象可以占据该容器	
Optimizer Hint	数据块	Forms Runtime为查询增加的一个提示字符串。查阅Oracle SQL参考手册和数据库管理员手册的文档，了解有关优化程序提示格式的细节	GS
Order By	数据块	追加到SQL SELECT语句上的一个字符串，Forms Runtime导致增加一个ORDER BY短语。可以在一个对查询出的数据进行排序的、用逗号分隔的项的列表中列出在数据块中的任意数据库项。可以把单词“ORDER BY”作为前缀，但这不是要求的	GS
Other Reports Parameters	报表对象	包含在报表的运行中的参数列表，缺省值为空白。列表的每个元素采用的格式是 <关键字>=<值>，<关键字>是任意的Reports Runtime命令行关键字	
Parameter Data Type	参数	参数的数据类型，可取值有：CHAR(缺省值)、DATE或NUMBER。参看Data Type属性可以了解详细内容	
Parameter Initial Value	参数	在表单启动时分配给参数的文字值。该值必须与Parameter Data Type属性相兼容	

China-pub.com
下载

(续)

属 性	对 象	描 述	Get/Set
PL/SQL Library Location	附属库	附属库的路径。如果该属性只包含库名，则Forms Runtime使用表单S60_PATH变量来解决对库进行引用的问题。如果该属性包含完整的路径，则Forms Runtime只在那一个位置查找。这个属性只是信息性的，不能对它进行修改	
PL/SQL Library Source	附属库	确定附属库是来自于一个文件 (File)还是来自于数据库 (Database)。这个属性只是信息性的，不能对它进行修改	
Popup Menu	项、画布	为画布或项显示的弹出菜单对象的名字。缺省值为NULL(无弹出菜单)	
Precompute Summaries	数据块	确定在对该块发出查询之前，是否要计算任何的汇总项的值，缺省值为NO。Forms Runtime针对汇总操作(计数、求和等等)发出一个专门的查询，选择该块中的所有记录并计算汇总值。如果该块包含有汇总项并且Query All Records属性是No，必须把这个属性设置为Yes。Forms Runtime两次激活Pre-Select触发器，一次正好在执行汇总查询之前，另一次在执行正常查询之前。Pre-Query触发器在专门的查询前只激活一次	
Prevent Masterless Operation	关系	确定用户在主块中没有主记录时是否能够查询或插入一个从属块的记录，缺省值是No。如果把这个属性设置为Yes，并且没有被查询的主记录，则用户不能够查询从属记录	GS
Previous Navigation Block	数据块	当焦点移动到以前的数据块时，Forms Runtime将导航的数据块的名字。缺省情况下，以前的数据块是在Object Navigator中出现的顺序中的前一个数据块。使用这个属性来改变导航的顺序，而不用重新排列Navigator中数据块的顺序。使用这个属性主要在运行时有计划地改变数据块导航的顺序	GS
Previous Navigation Item	项	当焦点移动到以前的项时，Forms Runtime将导航的项的名字。缺省情况下，以前的项是在Object Navigator中出现的顺序中的前一个项。如果希望有计划地在Form Builder中重新定义项的顺序而不用重新排列数据块中的项的次序时，设置这个属性	GS
Primary Canvas	窗口	Forms Runtime首先在窗口中显示的画布的名字，缺省值是NULL。如果导航到另一个画布中的项，Forms Runtime显示那个画布，而不是这个画布。如果使用Show_Window而不是通过标准的导航显示一个窗口的话，则仅仅需要设置这个属性。通常，应该使用导航来显示窗口	
Primary Key	除控制按钮、图表项和图像项以外的项	确定该项是否是数据块的基本表中的一个主关键字列，缺省值为No。该项必须是一个数据库项。数据块的Enforce Primary Key属性必须被设置为Yes。Forms Runtime将查询表，确保被标志为主关键字项的项组合在表中是唯一的。通常不要规定这个属性，相反应该在服务器上使用PRI-	GS

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
Primary Key	除控制按钮、图表项和图像项以外的项	MARY KEY约束条件，强制说明主关键字。当不能使用服务器约束条件时(例如正在使用一个不能增加约束条件的遗留下来的数据库)，这个属性提供强制约束的能力。参看Key Mode属性。如果把这个属性设置为Non-Updateable或Updateable，将需要把该数据块中至少一个项的Primary Key属性设置为Yes。另外，如果把DML Array Size属性设置为大于1的值，并且把Insert Allowed属性设置为Yes，则必须使用Enforce Primary Key和Primary Key属性指定一个主关键字，因为Forms Runtime在进行数组数据的维护时不能得到一个行识别符	
Program Unit Text	程序单元	单击More，显示一个程序单元包含在PL/SQL Editor中的PL/SQL代码	
Prompt	项	Forms Runtime为项显示的文本标签，缺省为空白。但是向导通常使用数据库的列名填充此项	GS
Prompt Alignment	项	确定如何把提示内容与项的边界进行排列，可取值有：Start(缺省值)、End或Center。Start是把提示内容排列在起始处。End是把提示内容排列在结束处。Center是把它放到中间的上方(通常用于多记录的数据块)	GS
Prompt Alignment Offset	项	确定与提示内容一起排列的项的中心点的距离，缺省值是0，表示把提示放到项的中间。如果对齐值是Center，这个属性把提示移到右边或左边；如果对齐值是Start或End，这个属性把提示向上或向下移动	GS
Prompt Attachment Edge	项	确定提示应该附加到项的哪一边，可取值有：Start(缺省值)、End、Top或Bottom	GS
Prompt Attachment Offset	项	与项边界之间的距离，该项是与提示对齐的项。缺省值是0，表示提示内容的右边靠近该项	GS
Prompt Background Color	项	确定该项提示的背景区域的颜色。缺省值是<Unspecified>，表示使用缺省的字体设置	
Prompt Display Style	项	确定提示内容显示在多记录块中的方法，可取值有：First Record(缺省值)或All Records。First Record为该块显示一次提示。All Records为块中的每一个记录都显示提示内容。当在一个多记录数据块中把记录分写在多行时，通常使用All Records这个值	GS
Prompt Fill Pattern	项	确定用于填充该项提示的背景区域的模式	
Prompt Font Name	项	显示Prompt属性中的文本的字体	
Prompt Font Size	项	项的提示信息的字体大小，以点为单位。缺省值是Unspecified，表示使用缺省的字体设置	
Prompt Font Spacing	项	项的提示信息的字符之间的距离，以点为单位。缺省值是Unspecified，表示使用缺省的字体设置	
Prompt Font Style	项	项的提示信息的字形，可取值有：Plain、Italic、Oblique、Underline、Outline、Shadow、Inverted、Overstrike、Blink。缺省值是Unspecified，表示使用缺省的字体设置	

China-pub.com
下载

(续)

属 性	对 象	描 述	Get/Set
Prompt Font Weight	项	该项提示信息的字体粗细，可取值有：Ultralight、Extralight、Light、Demilight、Medium、Demibold、Bold、Extrabold、Ultrabold。缺省值是Unspecified，表示使用缺省的字体设置	
Prompt Foreground Color	项	该项提示信息文本的颜色。缺省值是 <Unspecified>，表示使用缺省的字体设置	
Prompt Justification	项	确定在提示区域内的什么地方显示提示文本，可取值有：Start(缺省值)、End、Left、Right、Center。这个属性没有作用，因为不能设置文本区域的大小，因此它一直紧紧地包着显示文本	
Prompt Reading Order	项	显示该项提示信息的方向，可取值有：Default、Left to Right或Right to Left	
Prompt Visual Attribute Group	项	已命名的可视属性，用来为项的提示信息设置多种可视属性	
Property Class	所有对象	属性类的名字，该属性类是对象从设置的属性继承而来，缺省值是NULL(无子类)	
Query All Records	数据块	确定Forms Runtime在用户执行一个查询时，是否把一个查询结果集的所有记录取到数据块中，缺省值是No，表示只取出用Query Array Size属性指定的记录数。如果一个数据块包含有汇总项并且Precompute Summaries属性被设置为No，必须把Query All Records属性设置为Yes	GS
Query Allowed	数据块、除控制按钮、图表和图像以外的项	确定是否允许用户执行该块中的一个查询，或者该项是否能成为SELECT语句的一部分，这个SELECT语句是在用户执行一个查询时由Forms Runtime产生的。如果该项是参与关系的来自块中的外部关键字的一部分，Form Builder自动地设置这个属性为No。如果把该项的这个属性设置为Yes，还必须把该项的Visible属性设置为Yes。带有数据类型LONG的项从不参与一个查询	GS
Query Array Size	数据块	Forms Runtime在查询时用来从数据库中取出数据的数组的最大尺寸。数组尺寸越大，网络传输量越低，内存的使用率越高。如果有大容量的存储器和大量的查询以及许多用户，设置这个值大于1。每个用户的响应时间将会随着值的增大而有一点减慢，但是在网络上用数组进行存取，总体的响应时间将会减慢很多	G
Query Data Source Arguments	数据块	当Query Data Source Type属性被设置为Procedure时，这个属性包含有参数的名字、类型和值，这些参数传送给在Query Data Source Name属性中命名的用于查询数据的存储过程。Forms Runtime对Execute Query操作调用这个过程，而不是生成一条SQL SELECT语句。缺省值是NULL	
Query Data Source Columns	数据块	当Query Data Source Type属性被设置为Table、Sub-Query或Procedure时，单击在该属性上的More显示Query Data Source Columns对话框，该对话框包含有Forms Runtime把查询的结果集复制到的数据列的名字	

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
Query Data Source Name	数据块	当Query Data Source Type属性被设置为Table、Sub-Query或Procedure时，这个属性包含表名或子查询名，或执行Execute Query操作的存储过程名。Forms Runtime调用这个过程，而不是生成一条SQL SELECT语句。缺省值是NULL	
Query Data Source Type	数据块	确定Forms Runtime从该数据块得到数据记录的数据源的类型，可取值有：Table(缺省值)、Procedure、Transactional Trigger或FROM Clause Query。大多数数据块是基于一个表的。参看第6章中的“程序的封装”，了解有关使用查询过程的细节。如果把这个属性设置为Procedure或者是Transactional Trigger，那么不能使用数组存取(Query Array Size>1)或查询。然后选择使用On-Query触发器执行查询。也可以指定任何复杂的一个FROM短语作为数据源，Forms Runtime在生成的SELECT语句中使用FROM短语。这种方法类似于在服务器上定义一个视图并用它代替一个表作为数据块的基础，但是并不要求在服务器上真的定义视图	G
Query Length	文本项	确定用户在查询方式中能够输入的字符数量。当希望用户能够输入关系运算符(!=,>,<等等)来说明查询时，可以把这个属性设置成大于Maximum Length属性的一个值。这个功能扩展了查询操作，允许输入一个组合查询条件而不只是相等的比较式。最大值是255个字符	
Query Name	报表对象、图表项	在报表和图形显示模块中的查询对象的名字，代替在Data Source Data Block属性中说明的数据块。缺省是空白	
Query Only	项	确定在INSERT或UPDATE语句中是否包括该项，可取值有：Yes和No	GS
Radio Button Value	单选按钮项	当用户单击单选按钮时，所对应的Radio Group(单选组)项的值，缺省是空白	
Raise on Entry	画布	确定在用户导航到画布上的项时，将该画布显示在窗口的前面(Yes)，还是在当目标项在背面，有另一个画布已经显示在窗口中的时候才把该画布显示到前面(No,缺省值)	
Reading Order	显示和文本项	如果应用程序使用双向的民族语言支持功能，在一个单独的文本项内部同种语言词组的阅读顺序，可取值有：Default、Right-to-Left、Left-to-Right。阅读顺序控制着双向文本项的显示——文本项包含有罗马和本地两种语言的词组。如果文本项只有来自于单一语言的单词，这个属性无效	
Record Group	LOV、层次树	记录组的名字，对象从该记录中得到值	GS
Record Group Fetch Size	记录组	当Record Group Type属性的值是Query时，一次取到记录组中的记录的数量。缺省值为20。增加这个数量要使用较多的内存，但是如果记录组本身很大，并且有许多正运行该应用程序及正把记录查询到这个记录组的用户，能够减少总的网络传输量	

China-pub.com
下载

(续)

属 性	对 象	描 述	Get/Set
Record Group Query	记录组	若Record Group Type属性值是Query，则Forms Runtime用来聚集记录组的SELECT语句	
Record Group Type	记录组	确定是否用在Record Group Query属性中的SELECT语句来聚集记录组(Query)，还是用列的名字和值组成的对话框来构造该组	
Record Orientation	数据块	确定是按垂直格式(缺省值)还是水平格式来布局记录项。通常应该使用Frame对象来编排数据块的格式，而不是使用记录方向	
Relation Type	关系	确定在主从数据块之间的链接是一个 Join(缺省值，关系链接)还是一个REF(主块中指向从块中的引用行的一个REF列)。对值Join，使用Join Condition属性指定该连接。对值REF，使用Detail Reference属性指定REF项	
Rendered	显示和文本项	确定是否把该项作为被放弃对象显示，缺省值为Yes，如果设置这个属性为No，Forms Runtime在该项失去焦点时释放显示该项所要求的资源。使用这个设置能够减少系统资源的使用	
Report Destination format	报表对象	当Report Destination Type属性值是文件(在位图环境中)时使用的驱动程序，或者对字符方式环境的Report Destination Name中打印机的特征，可取值有：PDF、HTML、HTMLCSS、RTF、HTMLCSSIE、DELIMITED。PDF是Adobe Acrobat格式，使用Acrobat打印机驱动程序。HTML生成HTML 3.0的输出。HTMLCSS生成带有级连形式工作表的HTML 3.0。HTMLCSSIE相似于HTMLCSS，但是专指因特网浏览器。RTF生成Rich Text 表单at，许多标准的字处理软件能够读出此种格式。DELIMITED生成定界符格式的输出，适合于输入到电子表格或数据库程序中。不幸的是，没能找到在什么地方说明了用于这种选择的定界符	
Report Destination Name	报表对象	报表输出发送到的目的文件、打印机或电子邮件的名字。如果Report Destination Type属性值是File，则是一个文件名。如果Report Destination Type属性值是Printer，则是一个打印机名。如果Report Destination Type属性值是Mail，则是一个电子邮件的地址或者是一个分配的列表名。可以指定一个电子邮件名的列表，用带有圆括号的列表形式:(name,name,...,name)。缺省值是空白	
Report Destination Type	报表对象	希望生成的输出格式，可取值有：File(缺省值)、SCREEN、PRINTER、PREVIEW、MAIL、INTEROFFICE。值File将输出发送到在Report Destination Name属性中命名的、带有Report Formt中指定的格式的文件中。值Screen是输出发送到没有字体别名处理的一个预览中。 Printer把输出传递到在Report Destination Name属性中指定的打印机上。Preview与Screen相同，但是有字体的别名处理，对该输出当作页面描述语言输	

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
Report Destination Type	报表对象	出进行格式化处理。Mail把报表按照由Report Destination Name属性指定的电子邮件地址进行发送。InterOffice把报表发送给Oracle InterOffice的邮件用户，把该输出发送到InterOffice的储藏室中	
Report Server	报表对象	能够运行该报表的报表服务器，从服务器上的TNSNAMES.ORA文件中指定标准的服务器名字，例如repserver.world	
Required	列表和文本项	如果该项的值是NULL，确定在一个新的记录中是否认为该项是无效的，缺省值是 No。当用户视图从该项导航出去时， Forms Runtime报告一个错误并且拒绝进行导航，直到用户输入一个非NULL值为止。可以使用 Defer Required Enforcement属性延迟检验，直到离开该记录之前。相反地，如果把 Required属性设置为 Yes，可以为Initial Value属性设置一个有效值，保证在该项中一直有一个值。若弹出列表使得Required设置为No，Forms Runtime把一个额外的空白行追加到该列表中，允许用户输入一个 NULL值到该项中。对Tlist，如果Required被设置为yes，不能够取消选择当前值，只能选择另一个值	GS
Resize Allowed	窗口	确定是否允许用户重新定义窗口的大小，缺省值是Yes	
Rotation Angle	图形	旋转图形的角度值，缺省角是0	
Runtime Compatibility Mode	表单	确定是否使用表单 4.5或是Oracle Developer 6的性能。如果是在Oracle Developer/2000版本2或Oracle Developer 6.0中创建的表单，这个值是5.0，如果从Developer/2000版本1(表单 4.5)转换一个表单，该值是4.5。然后可以把它修改为5.0，把性能转换为Oracle Developer 6的性能，涉及到几件事情：有效性、项的初始化、日期的转换、设置Required属性以及处理在弹出列表和Tlist列表中的空值	
Savepoint Mode	表单	确定是否发出Oracle保存点(savepoint)，缺省值是Yes。保存点是Oracle的一个特征，它标记一个事务处理中的一个点，如果需要可以回溯到该点而不需要回溯整个事务处理。Oracle Developer在几个内部过程中使用保存点，例如象Post。如果是在一个非Oracle数据源上运行该表单，则可以通过把这个属性设置为No关闭保存点	GS
Scroll Bar Alignment	帧	确定是在帧的开始还是在结束(缺省值)的位置来显示多记录的滚动条	
Scroll Bar Canvas	数据块	画布的名字。在该画布上显示由 Show Scroll Bar属性允许的数据块的滚动条。缺省值是<Null>，表示不显示滚动条	
Scroll Bar Height	数据块	滚动条的高度	
Scroll Bar Orientation	数据块	确定把由Show Scroll Bar属性允许的滚动条是作为水平的还是垂直的滚动条进行显示，缺省是垂直的(Vertical)	

China-pub.com
下载

(续)

属 性	对 象	描 述	Get/Set
Scroll Bar Reverse Direction	数据块	确定滚动记录的方向是否反转，缺省值为No。只能对水平滚动条这样做，但是尽管那样，如果把这个属性设置为Yes，或许会使用户混乱	
Scroll Bar Visual Attribute Group	数据块	应用于滚动条以设置字体、颜色和模式属性的已命名的可视属性的名字	
Scroll Bar Width	数据块	滚动条的宽度	
Scroll Bar X Position	数据块	由Show Scroll Bar属性允许的该数据块的滚动条的水平X坐标，缺省值为0	
Scroll Bar Y Position	数据块	由Show Scroll Bar属性允许的该数据块的滚动条的垂直Y坐标，缺省值为0	
Show Fast Forward Button	声音项	确定该声音项是否显示快速前向控制，缺省值是No	GS
Show Horizontal Scroll Bar	画布、窗口、编辑器、图像项	确定是否对对象显示一个水平滚动条，缺省值是No。对一个窗口，还必须把Modal属性设置为No。窗口管理器必须支持水平滚动条	
Show OLE Popup Menu	OLE容器项	确定单击右键是否应该显示一个与OLE对象进行交互作用命令的弹出菜单，缺省值是 Yes	
Show OLE Tenant Type	OLE容器项	确定是否显示包含容器项的定义的OLE对象类型的边界，缺省值是Yes	GS
Show Palette	图像项	确定是否靠近该图像项显示一个图像操作调色板，缺省值是No。图像调色板提供三个命令：Zoom(单击图像增加它的大小)、Pan(使用一个抓取手形光标，在该项视图内部到处移动图像，部分图像可能是被隐藏的)、Rotate(通过单击它，把图像顺时针旋转90度)	GS
Show Play Button	声音项	确定声音项的控制是否显示Play按钮，缺省值为Yes。如果把这个属性设置为No，同时把Show Record Button属性也设置为No，则Forms Run-time无论如何都自动地显示Play按钮	GS
Show Record Button	声音项	确定声音项的控制是否显示Record按钮，缺省值为No。如果把这个属性设置为No，同时把Show Play Button属性也设置为No，则Forms Runtime无论如何都自动地显示Play按钮	GS
Show Rewind Button	声音项	确定声音项的控制是否显示一个反绕按钮，缺省值是No	GS
Show Scroll Bar	数据块	确定是否对数据块中的记录显示一个滚动条，缺省值是No	
Show Slider	声音项	确定声音项的控制是否显示滑动位置控制，以设置声音开始播放的地方，缺省值是 Yes	GS
Show Symbols	层次树	确定该树是否在每个分支节点的前面显示“+”或“–”符号，指示该节点是否是被部署的(–)或带子节点被折叠的(+)，缺省值是Yes	
Show Time Indicator	声音项	确定声音项的控制是否显示一个项指示器，显示用户播放声音的项的位置，缺省值是 Yes	GS
Show Vertical Scroll Bar	画布、窗口、编辑器、图像项	确定是否对对象显示一个垂直滚动条，缺省值是No。窗口管理器必须支持水平滚动条。对一个文本项，Multi-Line属性必须是Yes	
Show Volume Control	声音项	确定该声音项的控制是否显示一个音量控制，缺省值是Yes	GS

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
Shrinkwrap	帧	确定是否紧紧地包裹在帧之内的项，清除在帧边框和项边界之间的所有空白的空间，缺省值是No。不能调整一个Shrinkwrap帧的大小	
Single Object Alignment	帧	当Frame Alignment属性被设置为Fill时，一个单个对象在一条线上排列的方法，可取值为：Start(缺省值)、End、Center	
Single Record	数据块	确定是否强制Forms Runtime在一个控制块(不基于一个数据库表的一个数据块)中创建一个单个记录。在控制块中使用这个属性来处理汇总项、VBX控制项以及ActiveX控制项	
Sizing Style	图像项	确定当图像的尺寸不匹配项的尺寸(宽度和高度)时，进行处理的方法，可取值有：Crop(缺省值)或Adjust。Crop只显示适合图形的部分，剪裁边沿部分。Adjust按比例调整图像，使在图像之内完全地合适	
Sound Format	声音项	在数据库中存储声音的格式，可取值有：AU、AIFF、AIFF-C或WAVE(缺省值)	
Sound Quality	声音项	在数据库中存储声音所使用的品质级别，可取值有：Automatic(缺省值)、Highest、High、Medium、Low、Lowest	
Start Angle	弧形图形	弧的起始角度，以角度为单位，以水平坐标轴作为起点，缺省值为90	
Start Prompt Alignment	帧	当Layout Style属性被设置为表单时，把提示排列到项的水平边界的位置，可取值有：Start(缺省值)、Center、End	
Start Prompt Offset	帧	当Start Prompt Alignment属性值是Start时，提示信息与项之间的距离，以字符为单位，缺省值是0	
Subclass Information	所有被引用的对象	单击More，显示源模块的名字、源模块的类型(表单或菜单)和位置(文件系统或数据库)，以及在源模块中的源对象的名字	
Summarized Block	除开控制按钮、图表、图像、声音和VBX控制项以外的项	包含对因Calculation Mode属性设置为Summary的项进行汇总的数据的数据块，缺省值为空。如果有一个汇总项，该项包含一个基于这个数据块中的记录的合计值(计数、最小值等等)	
Summarized Item	项	对因Calculation Mode属性设置为Summary的项进行汇总的项，缺省值为空。不能对一个汇总项进行汇总	
Summary Function	项	用于按照Summarized Block属性对Calculation Mode设置为Summary的项进行数据汇总的函数，可取值有：Avg、Count、Max、Min、Stddev、Sum、Variance等函数。没有缺省值	
Synchronize with Item	除开OLE容器以外的项	当前时刻能够衍生它的值的项的名字。然后这两个项的值彼此相互镜像，当一个项的值改变，另一个项的值也改变	
Tab Attachment Edge	Tab画布	制表符将被关联到制表画布的位置，可取值有：Top(缺省值)、Bottom、Start、End、Left、Right	

China-pub.com
下载

(续)

属 性	对 象	描 述	Get/Set
Tab Page	项	当画布查阅到该项的Canvas属性是一个制表画布时，放置该项的制表页	G
Tab Page X Offset	Tab画布	画布的左边界和制表页的左边界之间的距离，以Coordinate System单位为单位	G
Tab Page Y Offset	Tab画布	画布的上边界和制表页的上边界之间的距离，以Coordinate System单位为单位	G
Tab Style	Tab画布	在显示制表符时所使用的形状，可取值有：Chamfered(缺省值)、Square或Rounded	
Title	LOV、窗口	在LOV或窗口标题栏上显示的标题，缺省值是NULL，表示显示缺省的名字(对窗口是对象的名字)。这个标题的允许长度取决于显示的分辨率(例如，800×600)	GS
ToolTip	项	当鼠标进入到一个项时，出现在该项下方的小方框中显示的文本，缺省值是空白	GS
Tooltip Background Color	项	工具栏提示的背景区域的颜色，缺省值是Unspecified	GS
Tooltip Fill Pattern	项	该项的填充区域的模式，缺省值是Unspecified	
Tooltip Font Name	项	Forms Runtime显示工具栏提示的字体的名字，缺省值是Unspecified，表示使用缺省的字体设置	GS
Tooltip Font Size	项	该项的工具栏提示的字体尺寸，以点为单位。缺省值是Unspecified，表示使用缺省的字体设置	GS
Tooltip Font Spacing	项	该项的工具栏提示的字符之间的距离，以点为单位。缺省值是Unspecified，表示使用缺省的字体设置	GS
Tooltip Font Style	项	该项工具栏提示的字体形式，可取值有：Plain、Italic、Oblique、Underline、Outline、Overstrike、Blink。缺省值是Unspecified，表示使用缺省的字体设置	GS
Tooltip Font Weight	项	该项工具栏提示的字体粗细，可取值有：Ultralight、Extralight、Light、Demilight、Medium、Demibold、Bold、Extrabold、Ultrabold。缺省值是<Unspecified>，表示使用缺省的字体设置	GS
Tooltip Foreground Color	项	工具栏提示文本的颜色，缺省值是Unspecified	GS
Tooltip Visual Attribute Group	项	Named Visual Attribute的名字，应用于该项的工具栏提示以设置字体、颜色和模式属性	
Tooltip White On Black	项	在单色显示方式下，确定是把工具栏提示显示为黑底白字(Yes)还是白底黑字(No)	GS
Top Prompt Alignment	帧	当Layout Style属性被设置为Tabular时，确定是否使项的提示于项的边排成一行，可取值有：Start(缺省值)、End或Center	
Top Prompt Offset	帧	确定当Layout Style属性被设置为Tabular时，从项的边界开始替换该项的提示信息有多少个字符单元，缺省值是0	
Top Title	编辑器	出现在编辑器标题栏上的文本	
Transactional Triggers	数据块	对一个非Oracle的数据源，确定包含事务处理触发器的块是否是一个事务处理的控制块。Base Table属性必须是NULL。把这个属性设置为Yes也就把Enforce Primary Key和Enforce Column Security属性设置成Yes了。如果使用的是Oracle，把这个属性设置为No	

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
Trigger Style	触发器	确定该触发器是一个PL/SQL触发器(缺省值)还是一个V2类型触发器。在版本6中这是只读的，并且不再容许有V2触发器	
Trigger Text	触发器	单击More，在PL/SQL Editor中显示触发器的PL/SQL代码	
Trigger Type	触发器	确定该触发器是内部的(缺省值)还是用户自定义的。内部触发器是Oracle Developer提供的触发器，用户自定义触发器是由用户命名的并由内部过程Execute_Trigger进行执行	
Update Allowed	数据块、除开控制按钮、图表和图像项以外的项	确定用户是否能够修改数据块中的任意项的值或者指定项的值，缺省值是Yes。设置的数据块的值取代在数据块的项中任何的设置	GS
Update Changed Columns Only	数据块	确定是否只对真正修改的那些项生成包含SET短语的一个UPDATE语句，缺省值是No。如果DML Array Size属性值是大于1的，则Forms Runtime忽略这个属性。当应用程序更新带有许多大量很少被更新的项(LONG、BLOB等等)的记录时，使用这个属性以减少网络的传输量。如果这种情形不是所描述的情形，不要把这个属性设置为Yes，因为失去了每次进行更新都重新使用被分析的UPDATE语句的能力。如果在对话中有许多更新的内容，则这种重新分析的总开销可能是值得注意的	GS
Update on Commit	图表项	确定当源块由于提交新的或更新的记录而发生改变时，是否更新这个图形显示，缺省值是Yes	
Update Layout	帧	确定何时更新该帧的布局：Automatically(缺省值)、Manually或Locked。Automatically值在对帧进行移动或调整大小或者修改帧的布局属性的任何时候更新该帧的布局。Manually值是在运行Layout Wizard以后或者在选择Update Layout按钮或菜单选项以后进行更新。Locked值从不进行更新	
Update Only if NULL	图像、列表、声音和文本项	确定是否不许修改该项，除非该项的值是NULL，缺省值是No，表示用户可以在任何情况下修改该项。Yes表示用户不能修改该项，除非它是NULL	GS
Update Procedure Arguments	数据块	当DML Data Target Type属性被设置为Procedure时，这个属性包含参数的名字、数据类型和值，该参数要传送到在Update Procedure Name属性中命名的用于更新数据的存储过程中。Forms Runtime调用这个过程进行Update Record操作，代替生成一个SQL UPDATE语句。缺省值时NULL	
Update Procedure Name	数据块	当DML Data Target Type属性被设置为Procedure时，这个属性包含执行更新一个记录的存储过程的名字。Forms Runtime调用这个过程，代替生成一个SQL UPDATE语句。缺省值时NULL。参看第6章的“程序的封装”一节，了解有关使用过程进行更新的细节	

China-pub.com
下载

(续)

属 性	对 象	描 述	Get/Set
Update Procedure Result Set Columns	数据块	当DML Data Target Type属性被设置为Procedure时，单击这个属性上的More，显示Update Procedure Result Set Columns对话框，该对话框包含一些列的名字，Forms Runtime把存储过程的结果集复制到这些列中。如果对存储过程的返回值进行更新，则应该这些列，例如一些新值或一个结果代码	
Update on Query	图表项	确定在源块通过查询进行修改时是否修改图形显示，缺省值是Yes	
Use 3D Controls	表单	确定是否在Windows下用三维方式显示表单，视觉和感觉是倾斜的，缺省值是Yes，这个属性禁止表单中的项的倾斜属性，每个项自动地显得降低了。如果对由Form Builder的一个早期版本的表单进行升级，这个属性被设置为No	
Validate from List	文本项	确定是否对该文本项进行校验检查，该文本项对应于由List of Values属性说明的LOV中的值的列表，缺省值为No。如果该值不匹配任何的列表元素，则Forms Runtime自动地显示该LOV	
Validation	表单	确定是否允许执行缺省的校验处理 (Yes缺省值)还是禁止执行它(No)。除了在希望避免进行有效性处理过程的短时期以外，不应该把这个属性设置为No。当通过一个触发器调用 Set_Form_Property又把该值设置为Yes时，进行有效性检查	GS
Validation Unit	表单	确定表单有效性检查的范围，可取值有：Default、Form、Block、Record、Item。参看第5章中“有效性检查单元和状态”关于有效性范围的完整解释。不同平台的Default的设置可以有不同的范围，例如，如果在一个块方式环境中可以是3270(应用程序属性User_Interface是BLOCKMODE)，其确认单元的缺省就是Block。大多数情况下它的缺省值是Item	GS
Value When Checked	复选框项	在用户对该复选框进行复选时的项值，无缺省值。该值必须与该项的数据类型兼容。如果一个程序单元对项赋值，其可视的复选框是被复选的	
Value When Unchecked	复选框项	当用户对该复选框不进行检查时的项值，无缺省值。该值必须与该项的数据类型兼容。如果一个程序单元对该项赋值，可视的复选框是不被检查的	
VBX Control File	VBX控制项	是VBX控件对应用程序可用的文件，无缺省值。避免指定路径	
VBX Control Name	VBX控制项	由VBX Control File属性指明的文件中的控件的名称，无缺省值	
VBX Control Value	VBX控制项	VBX控件的值属性 —— 该属性返回该项的值，如果是任意的，缺省值是缺省的VBX的值属性。如果没有，则缺省值是“value”	GS

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
Vertical Fill	帧	确定Layout Wizard是否可以在帧中使用空位置来安排对象，缺省值是 Yes。Yes表示Layout Wizard能够使用所有可用的空白位置，因此忽略Maximum Object Per Line属性。No表示Layout Wizard不能使用空白位置并把对象换到帧的下一行中	
Vertical Justification	图形文本	文本区域中文本的垂直位置，可取值有：Top (缺省值)、Center或Bottom	
Vertical Margin	帧	该帧的边界与内部的对象之间的垂直距离，缺省值为一个字符。这个属性控制定部和底部两个边界的距离	
Vertical Object Offset	帧	该帧中对象之间的距离，缺省值为 0	
Vertical Origin	图形文本	文本对象相对于它的原点的位置，可取值有：Top(缺省值)、Center或Bottom	
Vertical Toolbar Canvas	窗口	作为窗口中的一个垂直工具栏显示的画布，缺省值是NULL。该画布必须使得它的Canvas Type属性设置为Vertical Toolbar。在Windows中，如果使用表单 Vertical Toolbar Canvas属性显示该画布，则这个属性不起作用	
Viewport Height	画布	栈式画布的视图的高度，控制在运行时窗口中显示的画布的总数，缺省值为 0	S
Viewport Width	画布	栈式画布的视图的宽度，控制在运行时窗口中显示的画布的总数，缺省值为 0	S
Viewport X Position	画布	栈式画布的左上角相对于窗口的内容视图的左上角的X坐标，缺省值是 0	GS
Viewport X Position on Canvas	画布	视图的左上角相对于画布的左上角的 X坐标，缺省值是0。这是栈式画布的视图相对于栈式画布自身的位置，而不是相对于另一个画布上的视图的位置(Viewport X Position)	GS
Viewport Y Position	画布	栈式画布的左上角相对于窗口的内容视图的左上角的Y坐标，缺省值是 0	GS
Viewport Y Position on Canvas	画布	视图的左上角相对于画布的左上角的 Y坐标，缺省值是0。这是栈式画布的视图相对于栈式画布自身的位置，而不是相对于另一个画布上的视图的位置(Viewport Y Position)	GS
Visible	画布、窗口、制表页	确定画布或窗口是否最初或当前是可视的，缺省值是Yes。如果用户导航到画布上的一个项，在设置该属性为 Yes时，Forms Runtime使得该画布可视。可以使得窗口不可视，而与窗口中的画布的可视性无关。可以在可视的属性时使窗口完全地隐藏起来或者是图标化 /最小化。只有当视图的一些部分是可视的时，画布才是可视的。当栈式视图在窗口中内容视图的背后，或者一个单个的栈式视图完全地隐藏时，该栈式画布不是可视的	GS
Visual Attribute Group	所有可视的对象	设置对象的字体属性的方法，可取值有：缺省或一个命名的可视属性。缺省设置用缺省的颜色、模式和字体设置显示对象。如果指定一	

China-pub.com
下载

(续)

属 性	对 象	描 述	Get/Set
Visual Attribute Group	所有可视的对象	个命名的可视属性，则属性对象决定该设置。参看表16-1中的可视属性对象的描述，了解有关对象控制属性。参看第10章了解命名的可视属性和如何有效地适用它们的讨论	
Visual Attribute Type	可视属性	可视属性的类型，可取值有：Common(缺省值)、Prompt或Title。一个Common可视属性适用于大多数对象。也可以为项提示信息(Prompt)或帧标题(Title)建立专用的可视属性，选择后两种类型把属性的设置修改为Prompt和Title属性	
Where Clause	数据块	一个标准的SQL短语。当用户执行查询时，Forms Runtime将把它加到所产生的SELECT语句中。可以用WHERE关键字作为SQL表达式的前缀，但是它是可选的。可以从该数据块中引用数据块的列或者引用表单的参数(：PARAMETER <名字>)。不能嵌入SQL的注释。WHERE的最大长度是32KB	GS
White On Black	项、制表页、画布、窗口、单选按钮	确定对单色显示器是否以黑底白字方式显示对象(Yes)，或者不是(No)，缺省值是<Unspecified>	
Width	画布、项、编辑器、LOV、窗口	对象的宽度，以Coordinate System单位为单位，缺省值随对象的类型的变化而变化	GS
Window	画布	运行时在其内部显示该画布的窗口，缺省值是在Object Navigator中列出的第一个窗口节点	
Window Style	窗口	确定窗口是一个文档窗口(缺省值)还是一个对话框窗口。MDI应用程序在应用窗口结构内部显示文档窗口，而对话框窗口作为自由浮动的窗口显示在屏幕上	
Wrap Style	文本项、编辑器	确定当一行文本进行扩展，超过了该项或编辑窗口的宽度时，如何来显示文本，可取值有：Word(缺省值)、Character或None。Word对文本按最后一个完整的单词中断并换行到下一行上。Character对文本以这一行上的最后一个字符中断并换行，截断了该文本。None抑制所有内容的显示，而只显示文本的第一行	G
Wrap Text	图形文本	确定是否把文本对象中的文本换行到下一行，以使它适应文本框的边界，缺省值是Yes	
X Corner Radius	图形的圆角矩形	圆角矩形的边角的水平圆角的分量，以Coordinate System单位为单位，缺省值是10	
X Position	项、编辑器、LOV、窗口、画布	屏幕上对象的可视化表示法的左上角的水平位置(x坐标)，以Coordinate System单位为单位，缺省值是0。但是大多数情况下，该值取决于运行的各种向导的结果。可以通过显示LOV的项中的属性或者是通过显示其他画布的画布的属性，代替LOV和画布的位置	GS
Y Corner Radius	图形的圆角矩形	圆角矩形的边角的垂直圆角的分量，以Coordinate System单位为单位，缺省值是10	

China-pub.com 下载

(续)

属 性	对 象	描 述	Get/Set
Y Position	项、编辑器、LOV、窗口、画布	屏幕上对象的可视化表示法的左上角的垂直位置(y坐标)，以Coordinate System单位为单位，缺省值是0。但是大多数情况下，该值取决于运行的各种向导的结果。可以通过在显示该LOV的项中的属性或者是在显示其他画布的画布中的属性，代替LOV和画布的位置	GS

3. 格式掩码

下面给出了在项的表单 Mask属性中能够用于构造格式掩码的各种字符的引用方法。要嵌入字符，例如连字符或逗号，用双引号把该字符引起来。在写入数据库的文本中看不到这些字符出现，而当项从数据库中显示文本时，则把这些已嵌入的字符添加进来。通常可以使用FM前缀，容许用户输入带有或不带这些已嵌入字符的掩码。

1) 字符型格式掩码

在字符串的格式掩码中可以使用下列的字符：

- FM：允许用户输入一个任意长度的字符串，不用空格进行填充。不使用这个前缀表示希望用户精确地输入在掩码中说明的字符个数。
- X：任意的字母、数字或特殊的字符。
- 9：只能是数字字符。
- A：只能是字母字符。

2) 数值型格式掩码

- 9：显示数字字符，前导零作为空格。
- 0：显示前导零(前缀)或零值用0而不是空格显示(后缀)。
- $：插入一个美元符(用作前缀)。
- B：把零值显示为一个空格(用作前缀)。
- MI：在一个负数值后显示“ – ”(用作后缀)。
- PR：在尖括号中显示负数值，记帐方式(用作后缀)。
- , ：该位置上显示一个逗号，如果希望按国际惯例作为一个千位分隔符，则使用“ G ”字符(参见“国际化的格式掩码”)。
- . ：该位置上显示一个小数点，如果希望按国际惯例作为一个小数分隔符，则使用“ D ”字符(参见“国际化的格式掩码”)。
- EEEE：按科学记数法显示(格式必须准确地包含四个E)。
- FM：当作所键入的字符串接受，不向右对齐。

注意 对一个数值保持一个减号的位置。对正数，数值前保留一个空格。如果用户输入一个数值长于格式掩码，该项拒绝此次输入。但是，如果来自数据库中的一个数值长于格式掩码，该项使用舍入方法以适应掩码进行显示，但是保存它的真实值。然而，这样或许会使用户产生混乱，所以如果可能，尽量避免。

3) 日期型格式掩码

在日期型的格式掩码中可以使用下列字符：

- YYYY或SYYYY：四位年份，“ S ”表示“ BC ”日期的前缀，带有“ – ”号。

China-pub.com
下载

- YYY或YY或Y：年份的最后三位、两位或一位数字。
- Y，YYY：带有逗号的年份表示法。
- BC或AD：BC/AD指示。
- B.C.或A.D.：带句点的BC/AD指示。
- RR：两位年。从输入的日期推断世纪数，将输入的两位年份与计算机的内部时钟设置的年份和世纪进行比较。00~49年将被认为是21世纪(2000年)，而从50~99的年份将被认为是20世纪(1900年)。
- MM：月份(01~12，JAN=01)。
- MONTH：月份的名字，用空格填充到九个字符长度。
- MON：月份的名字，三个字符的缩写词。
- DDD：年的天数(1~366)。
- DD：月的天数(1~31)。
- D：周的天数(1~7，星期日=1)。
- DAY：日期的名字，用空格填充到九个字符长度。
- DY：日期的名字，三个字符的缩写词。
- J：儒略历的天数，从公元前4712年1月1日算起的天数。
- AM或PM：十二小时时钟表示法。
- A.M.或P.M.：带句点的十二小时时钟表示法。
- HH或HH12：一天的小时数(1~12)。
- HH24：一天的小时数(0~23)。
- MI：分钟数(0~59)。
- SS：秒数(0~59)。
- SSSSS：从子夜开始计时的秒数。
- /.,：在结果中被复制的标点符号。
- FM：允许用户输入的字符串，不要求精确的长度。
- FX：强制用户输入与格式掩码精确匹配的字符串。

注意 使用双引号允许加入其他的非支持的标点符号，斜线(/)、句号(.)和逗号(,)。

4) 国际化的格式掩码

在数值的格式掩码中使用这些字符，可以使得应用程序完全地国际化。Forms Runtime替代的真实值取决于在注册表或者Oracle初始化文件中的民族语言支持(NLS)功能的参数设置(例如，NLS_LANG)。

- C：作为前缀，增加国际通用的货币符号。
- L：作为前缀，增加本地的货币符号。
- D：在指定位置上增加小数点分隔符。
- G：在指定位置上增加千位分隔符。

16.1.2 菜单模块

图16-2给出了菜单模块结构的一个图示的概况。第2章中的“菜单模块”一节概述了菜单模块中的对象的结构和功能。

China-pub.com 下载

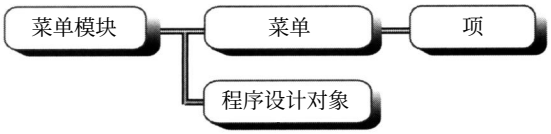


图16-2 菜单模块对象的层次结构

1. 菜单模块对象

表16-3列出了菜单模块的对象，并描述了它们在表单应用程序中的功能。

表16-3 菜单模块的对象

对 象	描 述
附加库	已经附加到其他模块上的PL/SQL库，这些库的目的是可以用变量代替程序单元
菜单	用户能够访问的一个文本项的列表，用单击文本域名行 (菜单名)的方法进行访问
菜单项	菜单上用户进行选择的文本域名
对象组	允许为以后进行复制或引用把可再用的对象打包的一个对象。它把一些对象的集合集中在一个单个的标题下
参数	一个可以进行赋值的对象，把它传送给一个模块的运行状态系统。该参数允许在运行时定制一个模块，增加它在不同情况下的可再用性
程序单元	一个PL/SQL程序包、函数或过程
属性类	集中了各种类型属性的一个对象
可视化属性	一个集中了一系列显示属性的对象，可以从另一个对象中对它进行引用。这个对象控制几个独立的属性：字体名、字体大小、字形、字间距、字体粗细、前景色、背景色、填充模式、字符方式逻辑属性和white on black

2. 菜单模块的属性

表16-4按字母字符的顺序列出了菜单模块中的所有属性。描述属性的同时，该表还列出了使用每个属性的对象。

表16-4 菜单模块的属性

特 性	对 象	描 述	Get/Set
Associated Menus	参数	参数申请使用的菜单的名字。当用户导航到这个菜单时，Forms Runtime提示用户在一个参数对话框中输入一个值	
Case Restriction	参数	参数中的文本的书写形式，可取值有：MIXED (缺省值)、UPPER或LOWER	GS
Checked	菜单项	检查形式的菜单项的状态，可取值有：CHECKED、UNCHECKED或空(缺省值)	
Command Text	菜单项	当用户选择该菜单项时所要执行的命令名。如果Command Type属性是MENU，命令文本是子菜单名。如果Command Type属性是PL/SQL，命令文本是一个无名的PL/SQL块	
Command Type	菜单项	执行命令的菜单项。Null不做任何操作，可以使用它来断开菜单项或者在准备增加代码前把它们捻熄，或者用作没有任何操作的菜单分隔项。Menu调用一个在Command Text属性中给出名字的子菜单。PL/SQL(缺省值)执行在Command Text属性中给出的一个PL/SQL块。其他选择(Plus、Current 表单s和Macro)是为了与以前版本相兼容，不要使用这些类型	

China-pub.com
下载

(续)

特 性	对 象	描 述	Get/Set
Conceal Data	参数	确定在用户把该参数输入到提示屏幕时是否要隐藏该参数的值，缺省值时No。可以使用这个属性设置一个菜单的口令参数	
Display without Privilege	菜单项	确定当用户是一个不具有对该项进行访问权限的任务成员时，是否要显示该菜单项。No(缺省值)表示Forms Runtime不显示该菜单项，而Yes表示它显示为暗灰色的菜单项	
Enabled	菜单项	确定Forms Runtime是否显示该菜单项(Yes,缺省值)还是把它变成暗灰色(No)	G
Fixed Length	参数	强迫用户对参数值提供Maximum Length属性指定的字符数量	
Help	菜单项	以字符方式，包含的该菜单项的帮助文本。当用户按下Help键选择该菜单项时，Forms Runtime显示这个文本	
Hint	菜单项、参数	在选择一个菜单项时，或者当对一个参数输入一个值时，所显示的文本	
Icon Filename	菜单项	包含为该菜单项显示的位图图标的图标文件名。不要加扩展名，使用UI60_ICON注册变量说明所有图标文件的路径。当Icon in Menu属性被设置为Yes时才发挥作用	GS
Icon in Menu	菜单项	确定是否在菜单项的文本旁边显示一个图标，缺省值是No	
Item Roles	菜单项	确定哪个菜单任务对一个菜单项已经进行了访问。显示一个任务列表，该任务列表已经在菜单对象的Module Roles属性中进行设置	
Keyboard Accelerator	菜单项	与菜单项关联的一个逻辑功能键，可取值为：ACCELERATOR1-ACCELERATOR5。使用Oracle Terminal来修改物理键与逻辑名的映射关系。也可以增加许多逻辑键。参看第15章有关的细节	
Label	菜单项、参数	菜单项的文本标签(它与项的名字不一样并出现在菜单中)，或者在参数对话框中对参数的值的提示	
Magic Item	菜单项	这个菜单项代表几个预定义的菜单项之一，可取值有：Cut(缺省值)、Copy、Paste、Clear、Undo、About、Help、Quit、Window。Forms Runtime以GUI平台的自然风格显示这个魔术项，带有对应的加速键。Form Builder提供Quit、Cut、Copy、Paste、Clear的功能。必须自己提供Undo和About的功能。Help项是一个子菜单项，所以Command Type属性必须是Menu，并且必须定义帮助选择的子菜单。Window项可能是一个子菜单，或者可能是NULL，在这种情况下，当用户选择它时该菜单项显示一个窗口的列表	
Main Menu	菜单模块	在该模块中的菜单对象的名字。当没有在表单模块中指定Initial Menu属性时，该属性值是Forms Runtime显示的缺省的菜单。如果设置那个属性为一个不同的菜单对象的名字，Forms Runtime显示那个菜单作为主菜单并且不允许用户查看不在该设计菜单下的其他菜单项	

China-pub.com 下载

(续)

特 性	对 象	描 述	Get/Set
Maximum Length	参数	该参数以字符为单位的最大尺寸，缺省值为 30	
Menu Description	菜单模块	标识该菜单模块的字符串，缺省是模块的名字。如果表单正运行在字符方式下，当用户导航到该菜单时，该字符串显示在信息行上	
Menu Directory	菜单模块	当在数据库中查找菜单时，这个属性标识出菜单模块的编译版本(.MMX文件)的位置，缺省为空。通常应该同意这个属性为空，并使用标准的表单S60_PATH注册变量来标识目录。Forms Runtime从Menu Filename属性中获取该文件名	
Menu Filename	菜单模块	当在数据库中查找菜单时，这个属性标识出菜单模块的编译版本(.MMX文件)的位置，缺省为该模块的名字。Forms Runtime从Menu Directory属性中获取该文件的路径	
Menu Item Radio Group	菜单项	当前单选菜单项所属的单选组的名字，无缺省值。对属于该组的所有单选菜单项必须指定相同的单选组名字。单选菜单项是希望在它们之间交替进行选择	
Menu Item Type	菜单项	菜单项的类型，可取值有：Plain(缺省值)、Check、Magic、Radio、Separator。Plain是一个标准的文本菜单项。Check是一个Yes/No Checked或Unchecked的菜单项。Magic是一个标准的窗口菜单选择(参看Magic Item属性了解细节)	
Module Roles	菜单模块	单击More，显示一个安全性任务的列表，从该列表中可以为这个模块中的菜单项拾取一个有效的任务。在每个项中，可以说明 Menu Item Roles属性来选择对该菜单项已经了访问哪个任务	
Parameter Menu Initial Value	参数	在表单启动时，Forms Runtime分配给该参数的值	
Required	参数	确定用户是否必须为该参数输入一个值，缺省值是No	
Secure	参数	确定是否隐藏用户对该参数输入的字符	
Share Library with Form	菜单模块	确定在当前激活的表单内是否能够共享在关联库中的任何程序包数据，缺省值是 Yes	
Startup Code	菜单模块	Forms Runtime 在装载菜单模块时执行的PL/SQL代码。虽然菜单不具备有触发器，但从本质上讲，这是一个When-New-Menu-Instance或Pre-Menu触发器	
Submenu Name	菜单项	在Command Type属性被设置为Menu时，对应于一个组合菜单的子菜单的名字，缺省值是 NULL	
Tear-off Menu	菜单	确定该菜单是否是一个Tear-off菜单，是用户可以拖动离开菜单显示区域的一种菜单，缺省值是No	
Use Security	菜单模块	确定Forms Runtime是否应该使用Menu Module Roles属性强制实施安全方案。在开发模块过程中可以把这个属性设置为No，然后在部署它之前把它设置为Yes。这样做会使得开发过程不受数据库作用的干扰	
Visible In Horizontal Menu Toolbar	菜单项	确定当前的表单是否应该把该菜单项作为一个图标显示在该表单的水平工具栏上，缺省值为 No	

China-pub.com
下载

(续)

特 性	对 象	描 述	Get/Set
Visible in Menu	菜单项	确定运行时菜单是否显示该菜单项，缺省值为Yes	GS
Visible in Vertical Menu Toolbar	菜单项	确定当前的表单是否应该把该菜单项作为一个图标显示在该表单的垂直工具栏上，缺省值为 No	

16.1.3 应用程序的属性

Get_Application_Property和Set_Application_Property内部子程序获取和设置各种的共用属性，这些属性在整体上应用到 Oracle Developer的表单应用程序。表 16-5列出了这些属性，并解释它们所做的哪些事情。

表16-5 表单应用程序的属性

特 性	描 述
Builtin_Date_Format	应用程序用于把字符串转换成日期的格式掩码。它应用于那些对用户是透明的字符串，大部分是在内部子程序的处理过程中
Calling_Form	来自于表单模块的名字属性的表单的名字，该名字被Call_form内部子程序调用，由当前的焦点处显示该表单
Connect_String	SQL*Net或Net8的连接字符串(例如，talbot/george@orc8)。通过用户名和口令属性，可以分别地访问连接字符串个别部分
Current_Form	正在运行的有焦点表单的可执行的文件的名字（.FMX)，是在表单的File_Name属性中的名字
Current_Form_Name	正在有焦点的表单的名字，是来自于表单的 Name属性中的名字
Cursor_Style	应用程序的鼠标游标的形状，可取值有：BUSY、CROSSHAIR、DEFAULT、HELP、INSERTION。Forms Runtime对长操作将显示BUSY游标，忽略这个设置，但是一旦BUSY游标被显示，就可以用其他的一些形状代替它
Datasource	数据库服务器的名字，可取值有：ORACLE(缺省值)、DB2、NULL、NON-STOP、TERADATA、NCR/3600、NCR/3700以及SQLSERVER
Display_Height	当前显示的高度，以Coordinate System单位为单位。可以用这个高度来计算窗口最佳的放置位置
Display_Width	当前显示的宽度，以Coordinate System单位为单位。可以用这个宽度来计算窗口最佳的放置位置
Operating_System	现行操作系统的名字，可取值有：WINDOWS、WIN32COMMON、UNIX、Sun OS、MACINTOSH、VMS以及HP-UX。其他值可能是有效的，查看专用平台的文档资料
Password	用户输入的口令。不能修改这个值。使用Connect_String应用程序属性，可以访问整个的连接字符串(用户名、口令和主机名)
PLSQL_Date_Format	如果不带显式的格式掩码使用To_Date函数，或者当把一个CHAR类型值赋值给一个DATE类型变量时，用于转换数据值的格式掩码。应该把这个格式设置为包含完整的世纪和时间格式，并且应该把这个掩码设置为与Buildin_Date_Format应用程序属性相同。还可以把一个日期格式作为 NLS注册变量进行设置，并通过可共用的应用程序的 User_NLS_Date_Format属性对它进行访问
Savepoint_Name	Forms Runtime能够进行回溯的保存点的名字。在 On-Savepoint或On-Rollback触发器中，可以获取这个名字，以便能直接地回溯到该保存点（一个非常平常的要求)。参看第5章，了解有关在 Post处理过程中的保存点行为的描述。可以使用表单的Savepoint Mode属性来完全断开保存点
Timer_Name	最接近到期的定时器的名字

China-pub.com 下载

(续)

特 性	描 述
User_Interface	当前窗口管理器的名字，可取值：BLOCKMODE、CHARMODE、MACINTOSH、MOTIF、MSWINDOWS、MSWINDOWS32、PM、WIN32COMMON、Web、X、UNKNOWN
User_NLS_Character_Set	由NLS_Lang注册表变量决定的当前的NLS字符集
User_NLS_Date_Format	当前的NLS日期格式掩码。可以使用这个属性来设置其他的应用程序的日期格式属性，例如PLSQL_Date_表单at
User_NLS_Lang	NLS_Lang注册表变量的完整的值，与User_NLS_Language、User_NLS_Territory和User_NLS_Character_Set属性相互联结
User_NLS_Language	NLS语言组件，来自与NLS_Lang注册变量
User_NLS_Territory	NLS区域组件，来自与NLS_Lang注册变量
Username	当前操作者的用户名。使用Connect_String属性可以获得被要求注册到数据库的整个的连接字符串

16.2 Report Builder

图16-3给出了报表模块结构的一个图形化的概述。第 2章中的“报表”一节概述了报表模

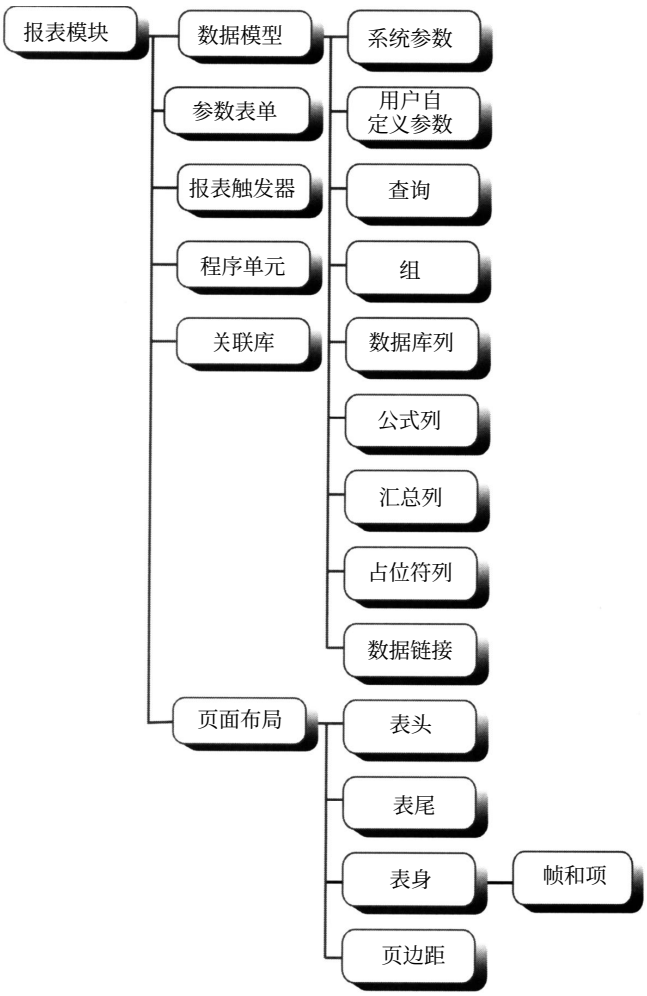


图16-3 报表模块的对象层次结构

China-pub.com
下载

块中的对象的结构和功能。

16.2.1 报表模块的对象

表16-6列出了报表模块的对象并描述了在表单应用程序中它们的功能。

表16-6 报表模块的对象

对 象	描 述
Anchor	把一个对象固定到另一个对象上的布局对象，保证第一个对象一直在对应于第二个对象的位置上。例如，可以使一个域固定到一个帧上，把该域放到帧内的中央位置
Chart Item	把一个Oracle Developer的图形显示模块显示在报表中的布局对象。图形显示的数据可能来自于显示模块或来自于报表
Column	数据模型组中的对象，与布局中的一个报表域相对应
Data Link	在主从关系的报表中链接主组与从属查询的对象
External SQL Query	包含可以导入到报表数据模型中的一个单个 SQL语句的模块
Field	包含一个值的布局对象
Formula Column	一个组中的特殊列，使用PL/SQL块对它进行计算，而不是直接从数据库获取数据
Graplic Object	几种布局对象之一，布局对象不包含值，取而代之是表示图片或图形的元素，可取值有：矩形、线、椭圆形、弧形、多边形、多线、圆角矩形或手画线。这些对象也被认为是样板图形
Group	标识多条记录的对象，这些记录是一个查询在报表中作为记录的一个重复组的返回值。在报表中有组的一个层次结构，反映记录彼此之间的嵌套关系。组还包含有一系列的域和列，它们构造出记录的结构
Group Frame	布局容器，它对应于一个组，并把组作为整体一齐进行显示
Layout Object	安置在报表布局上的几种类型的对象之一，可取值有：域、文本、图表项和图形
Parameter Form Boilerplate	一个文本、图形或图像对象放置在一个参数的表单中
Parameter Form Field	参数表单中的一个域(该表单是Oracle开发环境在运行时出现在用户面前用于收集参数的表单)
Placeholder Column	在一个组中定义的列
Query	一个SQL SELECT语句
Repeating Frame	对应于一个组中记录的重复序列的布局容器，并对该组中的每个记录显示一次
Section	报表中的一个头区段、主区段或尾部区段。每个区段包含一个主体和一个页边距
Summary Column	组中的一个特殊列，Developer对报表数据进行汇总计算，例如汇总求和或计数
System Parameter	一个Oracle Developer提供的变量，用户在运行时可以对它提供一个值
Template	包含报表内部各种帧和其他对象布局设置的一个对象，报表作为它的初始布局设置
Text Object	包含文本的布局对象，完全不同于包含来自数据模型中的一个值的域或文本域，这个对象也被当作是样板文本
User Parameter	一个用户提供的变量，用户在运行时可以为它提供一个值

16.2.2 报表模块的属性

表16-7按字母字符顺序列出了报表模块中的所有属性。在描述这些属性的同时，该表给出了使用每个属性的对象。

China-pub.com 下载

表16-7 报表模块属性

特 性	对 象	描 述
Additional Hyperlink Attributes	布局对象	超链接的附加HTML标记，通过Hyperlink属性定义
After Form Type	报表	当报表的目标文件是由Web浏览器预定的一个HTML文件时，在运行状态的参数表单底部显示的页脚的类型，可取值有：Text(缺省值)或File。Text表示显示在After Form Value属性中的字符串。File表示从由After Form Value属性命名的文件中获取文本或图形
After Form Value	报表	HTML文本，或者一个包含HTML文本的文件的有效文件名，或者在每页底部显示一个图形图标
After Page Type	报表	当报表的目标文件是由Web浏览器预定的一个HTML文件时，在每页的底部显示的页脚的类型，可取值有：Text(缺省值)或File。Text表示显示在After Page Value属性中的字符串。File表示从由After Page Value属性命名的文件中获取文本或图形
After Page Value	报表	HTML文本，或者一个包含HTML文本的文件的有效文件名，或者在每页底部显示一个图形图标
After Report Type	报表	当报表的目标文件是由Web浏览器预定的一个HTML文件时，在报表文档的底部显示的尾部的类型，可取值有：Text(缺省值)或File。Text表示显示在After Report Value属性中的字符串。File表示从由After Report Value属性命名的文件中获取文本或图形
After Report Value	报表	HTML文本，或者一个包含HTML文本的文件的有效文件名，或者在报表文档底部显示一个图形图标。可以使用这个属性把一个标准的链接或其他的素材放到报表底部
Align Summaries with Fields	模板	如果Style属性被设置为Form，则确定Report Builder是否把汇总域放到进行数据汇总的数据源域下，缺省值是 Yes
Alignment	模板	如果Style属性被设置为Form，则这个属性通知Report Builder如何在一页中安排标签项和域的位置，可取值有：Left(缺省值)、Right、Center、Flush、Column。Left把最左边的标签和域排列在页的左面，Right把最右边的标签和域排列在页的右边。Center把标签和域排列在页的中间。Flush把标签和域等距离的牌坊在页中，最左边标签或域排列到页的左边，而最右边的标签或域排列到页的右边。Column按照标签和列的最大宽度的大小以等距离方式排列标签和域
Application Command Line(PDF)	布局对象	当用户单击一个PDF(Adobe Acrobat)报表中的对象时要执行的命令。这个命令的执行代替了在Hyperlink属性中的操作
Background Color	模板	为布局对象的填充区域设置缺省的背景颜色。如果Fill Pattern属性是Transparent或Solid，则Form Builder不理睬这个属性
Base Printing On	布局对象	确定是否触发显示布局对象，取决于是 Anchoring Object(缺省值)还是Enclosing Object。这个属性与Print Object On属性协同操作，定义布局对象显示的精确触发。例如：Anchoring Object和Print Object On设置所有页在锚定的对象出现的每个页上触发显示
Before Form Type	报表	当报表的目标文件是由Web浏览器预定的一个HTML文件时，在运行参数表单的开始部分显示的页眉的类型，可取值有：Text(缺省值)或File。Text表示显示在Before Form Value属性中的字符串。File表示从由Before Form Value属性命名的文件中获取文本或图形

China-pub.com
下载

(续)

特 性	对 象	描 述
Before Form Value	报表	HTML文本，或者一个包含HTML文本的文件的有效文件名，或者在参数表单的开始部分显示一个图形图标。可以使用这个属性把一个图像或一个标准的链接放到参数表单的开始部分
Before Page Type		当报表的目标文件是由Web浏览器预定的一个HTML文件时，在每页的开始部分显示的页眉的类型，可取值有：Text(缺省值)或File。Text表示显示在Before Page Value属性中的字符串。File表示从由Before Page Value属性命名的文件中获取文本或图形
Before Page Value		HTML文本或包含HTML文本的具有有效文件名的文件，或在每页显示的图形图像
Before Report Type		当报表的目标文件是由Web浏览器预定的一个HTML文件时，在报表文档的开始部分显示的页眉的类型，可取值有：Text(缺省值)或File。Text表示显示在Before Report Value属性中的字符串。File表示从由Before Report Value属性命名的文件中获取文本或图形
Before Report Value		HTML文本，或者一个包含HTML文本的文件的有效文件名，或者在报表文档的开始部分显示一个图形图标。可以使用这个属性把一个标准的链接或其他素材放到报表文档的开始部分
Between Field and Labels (Horizontal)	模板	当Style属性是Form时，在一个域和它的标签之间水平距离数，缺省值是0，以报表的度量单位为单位
Between Frame and Fields (Horizontal)	模板	帧边界的边与所包含的域之间的距离数，缺省值是0。网格对齐在这里代替了精确的度量，因为当这个选项起作用时，域一直是与网格对齐的
Between Frame and Fields (Vertical)	模板	帧边界的顶部和底部与所包含的域之间的距离数，缺省值是0。网格对齐在这里代替了精确的度量，因为当这个选项起作用时，域一直是与网格对齐的
Between Master and Detail (Horizontal)	模板	主帧和从帧之间的水平距离数，缺省值是0。网格对齐在这里代替了精确的度量，因为当这个选项起作用时，域一直是与网格对齐的
Between Master and Detail (Vertical)	模板	主帧和从帧之间的垂直距离数，缺省值是0。网格对齐在这里代替了精确的度量，因为当这个选项起作用时，域一直是与网格对齐的
Between Page and Frames (Horizontal)	模板	页边界和最高级帧之间的水平距离数，缺省值是0。网格对齐在这里代替了精确的度量，因为当这个选项起作用时，域一直是与网格对齐的
Between Page and Frames (Vertical)	模板	页边界和最高级帧之间的垂直距离数，缺省值是0。网格对齐在这里代替了精确的度量，因为当这个选项起作用时，域一直是与网格对齐的
Between Sibling Frames (Horizontal)	模板	在同组级别上的帧之间的水平距离数，缺省值是0。对组的出身没有限制 (也就是说，可以由组树的不同部分构成组)。网格对齐在这里代替了精确的度量，因为当这个选项起作用时，域一直是与网格对齐的。
Between Sibling Frames((Vertical)	模板	在同组级别上的帧之间的垂直距离数，缺省值是0。对组的出身没有限制 (也就是说，可以由组树的不同部分构成组)。网格对齐在这里代替了精确的度量，因为当这个选项起作用时，域一直是与网格对齐的

China-pub.com 下载

(续)

特 性	对 象	描 述
Bookmark	布局对象	在HTML书签帧或Acrobat阅读器的内容区域的表中出现的一个链接，单击书签显示该布局对象。既可以单独地把书签名放入其中，也可以用一个分级数和英镑符号作为它的前缀。单独使用名字对书签根据对象在文件中出现的顺序进行排序。加上一个数字按照分级数对书签进行重新排序。如果把一个书签放到一个帧上，它自动地归诸于最靠近帧的左边角的可视的对象
Borders	模板	对象显示的边框类型，可取值有：All(缺省值)、Top Only、Bottom Only、Left Only、Right Only、Top and Bottom、Left and Right、Top and Right、Top and Left、Bottom and Right、Bottom and Left、All but Top、All but Bottom、All but Left、All but Right、None
Break Order	列	确定是否是按升序顺序(缺省值)或是按降序顺序显示这一列。这个属性应用于中断组列 —— 是对中断组识别不同值的列
Character Justification	模板	用于排列域的文本或字符类型数据源域的标签的方法，可取值有：Start(缺省值)、End、Left、Center、Right
Chart Column	图表	图形显示模块中的一个列或者一些列的列表，对应于Report Column属性中的列或参数。如果提供的列的列表是用逗号进行分隔的，Reports Runtime把报表的列映射到在列表中显示的每个列
Chart Filename	图表	显示模块文件的名字，该文件指明图形显示模块，扩展名.OGD视可选的。不要指定路径，使用注册表变量 REPORTS60_PATH来指定搜索路径
Chart Hyperlink	图表	对每个对象都可以指定的一个链接，和任何的超文本目标文件一起用，缺省值是空
Chart Parameter	图表	图形显示模块中的一个参数，希望通过Report Column属性中的报表列或参数进行设置
Chart Query	图表	图形显示模块查询的名字，从Report Column属性中查询的记录进行替换
Child Column	链接	子查询中的一个列的名字，链接到一个父组列上。这个列必须是一个数据库的列，不能是一个求和列或公式列，并且在这里不能使用字典索引。
Child Edge Percent	锚	向下或穿过锚的子对象的边框的百分比，指定在边框上锚的位置。这个值必须是0到100(含0~100)之间的一个整数
Child Edge Type	锚	一个对象的边框，在其上定位该锚，可取值有：Top、Bottom、Left、Right
Child Object Name	锚	被锚定的对象的名字(只读)
Child Query	链接	你定义的查询名作为在数据模型(只读)中链接到父查询的子查询
Collapse Horizontally	锚	如果父对象不进行打印，确定该锚是否应该是零宽度。这会引起子对象水平地移动到父对象可能出现的位置上
Collapse Vertically	锚	如果父对象不进行打印，确定该锚是否应该是零高度。这会引起子对象垂直地移动到父对象可能出现的位置上
Column Mode	重复帧	Reports Runtime如何获取和格式化在重复帧中的数据，缺省值是No。把这个属性设置为Yes，在完成以前的请求之前，强制格式化为开始，使得第二页的输出与第一页排列在一起。No设置禁止进行格式化，直到每个请求完成，使得请求不会排列在后续的页上。参看联机帮助，有关这个输出选项的图示例子

China-pub.com
下载

(续)

特 性	对 象	描 述
Column Type	列	列的类型，可取值有：Database-Object、Database-Ref、Database-Scalar、Database-Unknown(一个不被支持的列类型)、表单ula、Placeholder、Summary
Comments	对象	所希望的任意文本的文本域，不能超过64KB。
Compute At	汇总列	计算一个合计汇总列的百分数的组，可取值有：Report、Page、组名。缺省值Report表示对报表计算合计数。Page表示对每个页计算合计数。给出一个组名表示对该组计算合计数。换句话说，这里指定的组通过Reports Runtime除以该值得到一个合计值的百分数的方法指定合计数
Condition	链接	使父对象和子对象相互关联的SQL操作符，有：=(缺省)、<、<=、<>、>、>=、LIKE、NOT LIKE
Conditional Formatting	布局对象	单击More，显示Conditional 表单atting对话框，当条件为真时为对象设置输出的格式
Dash	模板	对象的破折号形式的线(下拉列表显示各种式样)，缺省值是实心线
Datatype	列、参数、参数表单域	值的数据类型，可取值有：Character、Date、Long、Long Raw、Number、Raw、Varchar、Varchar2。对数据库的列缺省值取决于它所对应的数据库的列的类型。对汇总列，该类型是在Source属性中的列的类型。对其他列，缺省值是Number。参数可以是Character、Number 或Date类型，缺省是Character类型。对参数的表单域，这个属性是在Source属性中指定的参数的数据类型，并且这个属性是只读的
Date Justification	模板	用来排列一个数据域的文本的方法，可取值有：Start(缺省值)、End、Left、Center、Right
Design in Character Units	报表	确定是使用字符单元网格(Yes)还是其他的度量单位(No，缺省值)来设计报表。这个属性与Include Borders属性一起使用按照字符方式显示对象的边界
Direction	报表	对整个报表的阅读方向，取值有：Left to Right或Right to Left，按照语言的阅读方向进行缺省设置。这个属性为报表中的版面对象提供缺省值，并且控制在报表编辑器中的版面方向
Disable Host Menu Item	报表	确定是否禁止在Runtime Parameter Form中的File|Host菜单项和Live Previewer，缺省是No
Disable Split Screen Key	报表	确定是否禁止Split Vertical和Split Horizontal功能键，缺省是No
Disable Zoom Key	报表	确定是否禁止在Live Previewer中的Zoom和Unzoom功能键，缺省是No
Display Name	图像	当光标移动到HTML或HTMLCSS中的图像上时，Reports Runtime显示的文本，缺省是空白
Distribution	区段	单击More，允许为该区段指定分布设置
Edge Background Color	模板	在Edge Pattern属性中定义的图案的背景颜色，缺省是白色。把Edge Pattern属性设置为透明的或实心的，使得Reports Runtime不理睬这个属性
Edge Foreground Color	模板	在Edge Pattern属性中定义的图案的前景颜色，缺省是黑色。把Edge Pattern属性设置为透明的或实心的，使得Reports Runtime不理睬这个属性
Edge Pattern	模板	用于对象边框的图案，由图案调色板进行设置，缺省是透明的

China-pub.com 下载

(续)

特 性	对 象	描 述
External Query Source File	查询	一个外部查询模块的模块名(在数据库中以DB:<名字>为格式的一个文件名或模块名)，用作报表数据模型的查询的源，省缺是空白。使用REPORTS60_PATH注册变量来指定路径，而不是把路径嵌入到文件名中
Fields Per Line	模板	对Style特性设置为Form的一个报表，可能在一行中出现的域的最大数目，省缺为0，表示没有限制
File Format	列	对Read from File特性设置为Yes的列，列值的格式，可取值有：Text(省缺值)、Image、CGM、Oracle Format、Sound、Video、OLE2、Image URL。Text是ASCII码格式的文本；Image是一个位图的图像；CGM是CGM格式的绘线型图形；Oracle Format是Oracle格式的绘线型图形；Sound和Video长原型格式的对象；OLE2是嵌入到报表中的OLE对象；Image URL是一个与图像的一个链接
Fill Pattern	模板	用于被对象包围的空间的图案，从图案调色板中进行选择。使用Foreground Color和Background Color设置来控制图案的颜色
Filler Type	组	用来在该组的记录上进行选择的类型，可取值有：None(无选择，省缺值)、First、Last、PL/SQL。First和Last设置取回在Number of Records特性中指定的记录数。First取回最开始的n条记录，而Last取回最后的n条记录。PL/SQL定义对每条记录返回True或False的一个PL/SQL函数，True表示包含有该记录，False表示不包含该记录
Font	样板	布局对象所使用的文本字体，省缺为Courier 10点阵的正常体。可以使用系统定义的任何字体，通过Font菜单和对话框可使用的字体
Foreground Color	样板	在Fill Pattern特性中定义的图案的前景颜色，省缺值是空白
Format Mask	域	在该域中显示日期和数字值的方法，省缺是空白。参看前面“格式掩码”一节，了解数字和日期格式掩码的语法。另外有一些可以在报表中使用的数字格式字符，有：()处理的数据两端打印圆括号；Y删除正负号；V用10N乘以该数，其中N是数字0、9、*和S记号，出现在V的右边；*打印除小数点左边是前导零以外的一个数字，当它答应一个星号时，并且也打印任何的尾部零到小数点的右边
Format trigger	布局对象	在对对象进行格式化之前执行的PL/SQL函数，这样就允许在运行时改变对象的格式化特性。该函数返回True或False。False表示执行当前对象来自于报表的情况，给出阻止显示所给出的一个条件值的方法。参看第五章“格式掩码”有关使用格式化触发器的细节
Function	汇总列	当计算该值时要执行的合计型运算，可取值有：Average、Count、First、Maximum、Minimum、%of Total、Std. Deviation、Sum、Variance
Height	报表、区段	参数Form页的高度，单位在Unit of Measurement特性中指定。对参数Form页而言，省缺值为4英寸/10厘米/300点；对区段，为区段的高度
Horiz.Space Between Frames	重复帧	在重复帧的例证对象之间水平方向插入的空白距离，单位在Unit of Measurement特性中指定，省缺值为0

China-pub.com
下载

(续)

特 性	对 象	描 述
Horizontal Elasticity	布局对象	如何改变对象的宽度，以适应其中的数据，可取值有：Contract、Expand、Fixed、Variable。Contract缩小该尺寸，但不超过他的原有尺寸；Expand则与之相反；Fixed保持原有的大小；Variable允许域进行扩展或收缩以适应数据。对一个图形，如果指定为Fixed，则Reports Runtime按比例调整图形以满足要求
Horizontal Panels per Page	区段	区段的逻辑页的宽度，兼顾物理页，省缺值为 1
Hyperlink	布局对象	当用户单击它时，显示一个文档内部位置的一个 URL，缺省值是空白。如果对一个帧指定为 Hyperlink，对该帧内部的所有的对象的该值都变成为超级链接，不持有它们自己的Hyperlink URL。对Adobe Acrobat PDF文档，应用程序命令行(Application Command Line)忽略该超级链接。如果对一个帧中的一个对象指定该特性，通过该链接用一个显示超级链接目标的完整窗口来代替该帧。为了只代替该帧，把Additional Hyperlink特性设置为：目标=<文件名>，其中<文件名>是进行代替的帧的名字。在当前文档之内的一个目标是按这种格式： #<目标>，其中 <目标>是在Hyperlink Destination特性中的目标的名字。在一个本地文档之中的一个目标是按这种文件格式： :/<路径>/<文件名>#<目标>。删除目标打开该文档而不到达其中的一个目标。在一个远程机器 (带有或不带目标)上的一个文档是一个简单的HTTP URL，可能包含#<目标>
Hyperlink Destination	布局对象	可以在Hyperlink特性中使用的唯一的名字，用以把一个对象通过一个超级链接链接到这个对象上。该名字只能使用ASCII字符、数字、下划线。如果对象是一个帧，该链接被链接到最靠近帧的左上角的对象上
Icon Name	按钮	对Label Type特性设置为Icon的按钮，时想要在按钮上显示的图标的文件名，省缺为空白。该文件必须放在 UI_ICON注册变量说明的目录中
Image	样板	用于替代样板中外部样板对象所引用的图像的文本文件
Include Bitmapped Objects	报表	确定当报表在字符方式下运行时是否把所有的位图对象转换到方框中，省缺值为No，表示删除这些对象，不显示方框
Include Borders	报表	在字符方式中对任何对象都突出边框，边框线的宽度大于0，缺省值为No，与Design in Character属性一起使用这个属性，告知Report Builder显示边框
Initial Value	参数	参数的缺省值。确定的系统参数有指定的缺省值：DESTYPE=Screen、DES表单AT=dflt、COPIES=1，否则缺省值是空白。该值必须域参数的Datatype属性中所给出的类型相容
Input Mask	参数	Reports Runtime用于检查用户通过命令行或参数表单输入值有效性的格式掩码。参看表单 at Mask属性
Inter-Field(Horizontal)	样板	一个组中的各域之间的水平距离数，单位在Unit of Measurement属性中指定。对Style属性设置为Form的报表，该属性是在一个域和标签混合的结尾与下一个域之间的距离数。对象是按网格对齐的，所以确切的距离可能是不同的
Inter-Field(Vertical)	样板	一个组中的各域之间的垂直距离数，单位在 Unit of Measurement属性中指定，缺省值为 0。对象是按网格对齐的，所以确切的距离可能是不同的
Inter-Frame(Horizontal)	样板	一个组中的各帧之间的水平距离数，单位在 Unit of Measurement属性中指定，缺省值为0.0087英寸或等量的

China-pub.com 下载

(续)

特 性	对 象	描 述
Inter-Frame(Vertical)	样板	一个组中的各帧之间的垂直距离数，单位在 Unit of Measurement属性中指定，缺省值为0.0136英寸或等量的
Justify	样板	在标题中对齐文本的方法：Start(缺省值)、End、Left、Center、Flush、Right
Keep with Anchoring Object	布局对象	确定是否把该对象与它的锚定对象放在同一页上，缺省值为No。如果该对象是一个重复帧，则把该帧与它的对象放在一起。该锚可能是一个隐含的锚
Label Type	按钮	按钮的类型，可取值是：Text(缺省值)或Icon。使用Icon Name属性来指定图标文件
Line Stretch with Frame	图形线	线的关联帧的名字，把该线的锚定在帧的结束位置，缺省值为NULL。当报表格式化时，该线与帧一起延伸。帧必须完全地包含该线，并且必须是水平线或垂直线，而不能是斜线
List of Values	参数	单击More显示List of Values对话框，该对话框允许为输入的参数值创建一个值列表。可以检查 Hide First Column复选框在用户对它设置该值时把一个数据列隐藏起来，或者可以检查Restrict List to Predetermined Value以阻止用户输入不在该列表中的一个值
Max.Horizontal Body Pages	报表	可能出现在报表中的物理页宽度范围内的表身的页的最大数，缺省值是10
Max.Vertical Body Pages	报表	可能出现在报表中的物理页高度范围内的表身的页的最大数，缺省值是10
Maximum Records Per Page	报表	Reports Runtime在一个逻辑页上可以进行格式化的重复帧的最大数，缺省为空白，表示Reports Runtime可以把尽可能多的重复帧放在该页上
Maximum Rows to Fetch	报表	Reports Runtime对查询将取出和显示的最大行数。可以使用这个属性在报表开发过程中来限制较大的查询。汇总计算忽略没被取出的行
Minimum Widow Lines	模板、域	在Reports Runtime开始打印该对象的逻辑页上必须出现的最小线数，缺省值为0，表示打印合适的文本
Minimum Widow Records	重复帧	在Reports Runtime开始打印该对象的逻辑页上必须出现的最小帧数，缺省值是空白，无最小值。这确保在每个页上至少具有这些记录数
Multimedia Column	按钮	当用户按下该按钮时，运行的报表中的一个数据库列的名字，缺省值为空白
Multimedia Column Type	按钮	在Multimedia Column属性中指定的多媒体对象的格式，可取值有：Image、Video、Sound。缺省值取决于数据库列的类型
Multimedia File	按钮	包含在用户按下该按钮时Reports Runtime执行的一个多媒体对象的文件的名字。使用 REPORTS60_PATH注册变量来设置路径，而不要把路径嵌入在文件名中
Multimedia File Type	按钮	在Multimedia File属性中的文件的格式，可取值有：Image(缺省值)、Video、Sound
Name	对象	该对象的名字，长度不超过30个字符。Report Builder使用缺省的命名方案创建这些名字，用前缀表明对象的类型，例如G_表明是组
Number Justification	样板	当Source是数值型列时，用来排列域文本的方法，可取值有：Start(缺省值)、End、Left、Center、Right
Number of Pages	报表	在Runtime Parameter Form中页的总数

China-pub.com
下载

(续)

特 性	对 象	描 述
Number of Records	组	在由Filter Type属性的值First或Last限制的一个组中包含的记录数，缺省值为0，表示所有的记录
Orientation	区段	确定是按横向还是纵向方向输出该区段，通过系统参数ORIENTATION或者Page Setup对话框的设置来设置缺省值
Page Break After	布局对象	确定是否把锚定到这个对象的所有的子对象移动到下一页，缺省值是No。这个属性不会影响到同级对象，同级对象不会被锚定到另一个对象上
Page Break Before	布局对象	确定是否把该对象放到Reports Runtime开始进行格式化的页后面的页上，缺省值是No。这不会影响到这页上的下级对象但又没有以任何方式锚定到该对象的那些对象
Page Navigation Control Type	报表	确定Page Navigation Control Value属性是什么：Text(缺省值)或文件
Page Navigation Control Value	报表	HTML命令字符串或包含这样的命令的文件名，这些命令在HTML页流报表输出中实现导航按钮功能。编写自己的小型程序代码，必须增加两个变量： &TotalPages和&file_name(输出目标文件的名字)
Page Numbering	域	对带有页计数类型的域，单击More显示Page Numbering对话框，其中允许设置对该域进行页数计算的方法
Page Protect	布局对象	确定是否要试图把整个对象和它所包含的所有内容都放在同一个逻辑页上，缺省值是No。如果该对象不能完整地符合开始进行格式化的页，该对象移动到下一页。如果允许一个页几乎或完全是空白的，Reports Runtime不会应用该属性
Panel Print Order	报表	打印组成一个逻辑页的物理页的顺序，缺省值是：Across/Down，表示先从左向右，再从上向下地打印这些页，或者是Down/Across表示先从上到下，再从左向右地打印这些页
Parent Column	链接	与子查询中的一个列相互关联的父组中的一个列 (只读)
Parent Edge Percent	锚	该锚在父对象的边框上的位置，根据百分比数，或者穿越在Parent Edge Type属性中指定的边框
Parent Edge Type	锚	该锚挂接到其父对象的边，可取值有： Top、Bottom、Left、Right
Parent Group	链接	在建立该链接时作为父组定义的组的名字 (只读)
Parent Object Name	锚	该锚关联到的父对象的名字 (只读)
PL/SQL Filter	组	单击More显示PL/SQL Editor，对该组的过滤器函数进行修改。当这个属性返回值为假时， Reports Runtime不显示该组。必须把Filter Type属性设置为PL/SQL
PL/SQL Formula	公式和占位列	单击More，显示PL/SQL Editor，把对该列的值进行赋值的PL/SQL表达式放到这个Editor中
PL/SQL Statement	查询	单击More显示PL/SQL Editor对一个函数进行编辑，该函数通过一个参考游标为查询返回数据，而不是依靠一条SQL语句
PL/SQL Trigger	按钮	当用户单击在Live Previewer中的按钮时执行的过程
Place Labels Above Fields	样板	当Style属性被设置为Form时，确定是否把域的标签放到所属域的上方，而不是其左边，缺省值为No
Previewer Hint Line Text	报表	当把Use Previewer Hint Line属性设置为Yes时，在Live Previewer中作为第二到最后一行显示的文本
Previewer Status Line Text	报表	当把Use Previewer Status Line Text属性设置为Yes时，在Live Previewer的最后一行上显示的文本
Previewer Title	报表	在Live Previewer的定部出现的文本，缺省值是NULL，表示使用缺省的预览标题

China-pub.com 下载

(续)

特 性	对 象	描 述
Print Direction	重复帧	显示连续的重复帧实例的方向，可取值有： Down(缺省值)、Across、Across/Down、Down/Across。拾取这些选择之一允许实现各种类型的向下和交叉的报表格式(参看第7章)
Print Object On	布局对象	对象在报表中出现的频率，可取值有：All Pages、All but First Pages、All but Last Pages、Default、First Page、Last Page。这个属性与Base Printing On属性相互有关系
Printer Code After	布局对象	在一个字符模式的报表中对象的每一行之后执行的打印机转移字符序列。格式是&number，其中number是在打印机定义的文件中分配的一个序列的数字
Printer Code Before	布局对象	在一个字符模式的报表中对象的每一行之前执行的打印机转移字符序列。格式是&number，其中number是在打印机定义的文件中分配的一个序列的数字
Product Order	汇总、公式、和占位列	在矩阵报表中对汇总列、公式列和占位列的组进行赋值的顺序。如果汇总列的 Reset At属性未被设置为 Report，Product Order属性必须是它的源列的 Product Order属性的子集。该汇总列的 Product Order属性应该是它的源列的 Product Order属性的子集。如果 Reset At属性被设置为 Report，则Product Order属性必须是源列的Product Order属性的一个前缀
Read from File	列	确定对该列是从一个文件(Yes)还是从数据库(No)中获取图像或文本值。使用File Format属性指定文件的格式
Report Group	图表	在报表中由图形显示程序汇总的组
Report Height	区段	报表页的高度，以字符为单位，缺省值为 80
Report Width	区段	报表页的宽度，以字符为单位，缺省值为 66
Reset At	汇总列	将汇总列的值复位位零或空白的组。 Reset At可能是Report、Page或一个组的名字。 Report对整个报表计算一个汇总列。 Page对每一页计算该列。 Group在每组的记录上计算该汇总列
Role Name	报表	单击More按钮，显示Set Role 对话框。文本是数据库的安全性作用的名字，用于在运行时刻运行该报表。设置这个作用允许用户运行该报表，这些用户对在报表数据模型的查询中的表不具备访问权限的
Set Break Order	列	确定是否设置根据Break Order属性显示该列值的顺序，缺省值是Yes
Source	汇总列、重复帧、参数表单域、域	进行汇总的列名，包含在帧中重复出现的数据元素的组名，参数表单域的参数名，该域从其中获取值的列、参数或特定值。特定值包括： &Current Date、&Logical Page Number、&Panel Number、&physical Page Number、&Total Logical Pages、&Total Panels、&Total Physical Pages。对页号，可以使用Page Numbering属性来控制页号如何进行编排
Source Datatype	域	该域的数据源的数据类型(只读)
Source File Format	模板	对链接到一个外部文件的样板，该文件的格式，可取值有：Text、Image、CGM、Oracle Drawing Form at、Image URL
Source Filename	模板	链接该样板的外部文件的名字。应该使用 REPORTS60_PATH来指定路径
SQL Clause	链接	关联父和子的SQL的类型，有：HAVING、START WITH、WHERE。如果指定HAVING，则子查询的SELECT语句必须有一个GROUP BY短语；如果指定START WITH，它必须有一个CONNECT BY 短语

China-pub.com
下载

(续)

特 性	对 象	描 述
SQL Query Statement	查询	实现该查询的SQL SELECT语句
Start in Zoom	报表	确定是否不带边框地显示Live Previewer，只给出报表的输出，缺省值为No
Style	样板	确定该组是具有Tabular还是Form风格的页面布局。在Tabular页面布局中，标签在域的上方或在重复帧的外边。在Form页面布局中，标签在域的左边或在重复帧的里边。带有子组的Group Above和Form-Like页面布局使用表单风格；而Tabular、Group Left、不带子组的Group Above、Matrix和Matrix Breakall使用Tabular风格
Suppress Previewer Title	报表	确定是否在字符模式的预览中禁止标题，缺省值为 Yes
Text	按钮、样板	出现在Label Type属性值为Text的按钮上的文本标签(居中对齐)，或者是在样板中显示的样板文本对象的一个文本替换文件的名字
Text Color	样板	对象文本的颜色，缺省值为黑色
Type	模板、按钮、查询	样板的格式(只读)，可取值有：按钮的类型(在Multimedia File属性中命名的多媒体文件，缺省值；多媒体列，源于Multimedia Column属性的一个数据库的列；或在 PL/SQL Trigger属性中的PL/SQL)或者查询类型(只读)
Unit of Measurement	报表	Report Builder用于对在报表中的各种对象指定大小的公制单位，可取值有：Inch(缺省值)、Centimeter、Point
Use Previewer Hint Line	报表	确定是否显示在Previewer Hint Line属性中的文本，缺省值为No
Use Previewer Status Line	报表	确定是否显示在Previewer Status Line属性中的文本，缺省值为No
Use Vertical Spacing	样板	确定是否移动对象以占据在前面行上未使用的空间，缺省值是Yes
Validation Trigger	参数	单击More，显示PL/SQL Editor，编写对参数进行有效性检查和/或进行修改的程序块
Value If Null	列	替代NULL的值，缺省为空白，表示不进行替代。如果在一个中断列上设置这个属性， Reports Runtime一次性地取出在该组中的所有的记录行，可能会降级执行。在这种情况下，在查询中使用NVL函数来对NULL替换值
Vert.Space Between Frames	重复帧	在重复帧的实例之间的垂直距离，单位在Unit of Measurement属性中指定，缺省值为0
Vertical Elasticity	布局对象	如何改变对象的高度，以适应在其中的数据，可取值有：Contract、Expand、Fixed、Variable。Contract减少尺寸但扩展尺寸不超过原始尺寸；Expand则相反；Fixed保持同样的高度；Variable使域进行扩展或收缩以适应数据。对一个图形，如果指定为Fixed，则Reports Runtime按比例调整图像的大小
Vertical Panels per Page	区段	在物理页中区段的逻辑页的高度，缺省值是 1
Visible	域	确定是否显示该域，缺省为Yes。可以引用在样板文本中的隐藏域中的值，使用语法 &name或&<name>，用尖括号把在其他文本中的值括起来，而不追加到开头和结尾的位置上
Width	列、参数、参数表单、报表、区段	一个列或参数可以显示的文本的数量，以字符为单位；参数表单页的宽度，单位在 Unit of Measurement属性中指定；报表的物理页的宽度；区段的宽度

China-pub.com 下载

16.3 Graphics Builder

图16-4给出了图形显示模块结构的一个图形化的概述。第 2章中的“图形”一节概述了显示模块中的对象的结构和功能。

图16-4 图形显示模块的层次结构

16.3.1 显示模块的对象

表16-8列出了显示模块的对象，并描述了在一个表单应用程序中的功能。

表16-8 图形显示模块的对象

对 象	描 述
Layout Object	图表的图形元素
Parameter	一个变量，通过命令行或通过从一个表单或报表将参数传送过来的方法进行设置
Query	一个SQL SELECT语句，定义图表的基础的数据集
Sound	表示声音数据的一个对象
Template	为图表定制的选项集
Timer	作为图形显示的一种报警时钟的一个对象

16.3.2 显示模块的属性

Graphics Builder与其他的两个主要组件， Form Builder和Report Builder，有所不同。在Graphics Builder中没有Property Palette和通过Object Navigator进行设置的属性的长列表。取而代之的是，每个对象都有一个对话框，用来对它的属性进行维护。表 16-9按字母字符的顺序列出了在这些对话框中的所有属性。在描述属性的同时，该表也给出了使用这个属性的对象，也就是使用属性的对话框。

China-pub.com
下载

表16-9 图形显示模块的属性

特 性	对 象	描 述
Active	定时器	确定定时器当前是否是活跃的(也就是，当显示执行时定时器被激活还是不被激活
Active Layer	层	当前活动层的名字
Begin	图表	如果Mapping Type属性值是Gantt，则作为开始值进行绘制的列
Chart Category	图表	在分类(x)坐标轴上绘制的列
Chart Values	图表	如果Mapping Type属性值是General，在值(y)坐标轴上进行绘制的列
Close	图表	如果Mapping Type属性值是High-Low，作为关闭值进行绘制的列
Custom Execute Procedure	查询	对定制查询，进行调用的PL/SQL过程。只有那些具有一个有效的定制查询触发器过程头的过程出现在这个属性的下拉列表中
Data Range	图表	图表进行绘制的数据行，可取值有：Plot All Rows (缺省值)或Plot Rows(只对那些落在指定范围内的数据行)
Data Type	声音	声音文件的类型，可取值有：8 Bit Mono、8 Bit Stereo、16 Bit Mono、16 Bit Stereo
Date Format	查询	Oracle的日期格式，Graphics Runtime用来识别来自一个文件的日期数据(参看File属性)
Device	声音	录制小或回放声音的设备
Encoding	声音	声音编码的类型，Signed/2's Complement PCM是唯一的选择
End	图表	如果Mapping Type属性值是Gantt，作为结束值进行绘制的列
Event Types	图表	调用在Procedure属性中的按钮过程的鼠标事件，可取值有：Mouse Button Down、Mouse Button Up、Mouse Move with Button Down、Mouse Move with Button Up(或者是这些被选中事件的任意组合)。Button Up只适用于层
Execute On	查询	何时自动地执行该查询，可取值有：Opening Display (缺省值)或Timer(在一个指定的时间间隔出发一个定时器)
Execute Query	布局对象	当用户选择对象时执行的查询
Existing Layers	层	在显示模块中层的树
Field Template	图表	如果Mapping Type属性值是General，对在Chart Value列表中选择的列的结构进行设置的样板
File	查询	从中获得该查询的文件的名字
Filter Function	图表	一个PL/SQL函数，对包含有来自在Query属性中的查询的记录时返回True，而对不包含有记录时返回False
Format Trigger	布局对象	在Graphics Runtime对对象进行格式化时执行的过程的名字。只有那些具有一个有效的定制查询触发器过程头的过程出现在这个属性的下拉列表中
High	图表	如果Mapping Type属性值是High-Low，作为高的值进行绘制的列
Initial Value	参数	在Graphics Runtime创建一个参数变量时，对该参数变量所赋的值
Interval	定时器	以秒和分为单位的时间数，是在执行Procedure属性中的PL/SQL过程之间传递的定时器的时间数
Low	图表	如果Mapping Type属性值是High-Low，作为低的值进行绘制的列

China-pub.com 下载

(续)

特 性	对 象	描 述
Mapping Type	图表	Values属性页的页面布局，可取值有：General(缺省值，单个值的一个列表)、Gantt(开始和结束值)、High-Low(高、关闭和低的值)
Maximum Number of Rows	查询	当New Data属性是Appends to Old Data时，该查询可能返回的数据行的最大的数量， Graphics Runtime丢弃任何额外的数据行
Name	布局对象、图表、查询、参数、定时器	对象的名字
New Data	查询	如何处理最新的检索数据：Replaces Old Data表示丢弃所有以前的数据， Appends to Old Data表示保留以前的数据并增加新的数据
New Name	层	当单击Rename按钮时，用于重新对该层进行命名的名字
Parameter	参数	在参数对话框中修改的参数的名字
Post-Execution Trigger	查询	在查询执行以后调用的PL/SQL过程。只有那些具有有效的post-query触发器的过程头的过程才在这个属性的下拉列表中出现
Procedure	布局对象、图表、定时器	对象或图表的PL/SQL按钮过程的名字。当在Event Types属性中指定的一个鼠标事件出现时，该过程执行。对定时器，该过程是当定时器在指定的时间间隔激活时要执行的过程。只有那些具有一个有效的按钮或定时器触发器的过程头的过程才出现在下拉列表中
Query	图表	一个SQL SELECT语句，该语句返回作为图表的基本数据使用的数据
Query Columns	图表	可以用来作为图表分类(x坐标)和图表值(y坐标)的列
Rate	声音	进行声音回放的速率，以kHz(千赫兹)为单位
Save Old Data	查询	当New Data属性是Replaces Old Data时，确定是否要存储在缓冲区中的以前检索到的数据，这些数据可以用PL/SQL内部子程序进行访问
Set Parameter	布局对象	当对象被选中时，其值被设置的参数
SQL Statement	查询	该查询要执行的SQL SELECT语句的文本
Subcategory	图表	在一个断开图表中用作一个子分类的列
Subtype	图表	在Type属性中选择的类型内部的图表的特殊类型
Template	图表	用于构造该图表的图表样板
Timer	定时器	在Timer(定时器)对话框中进行修改的定时器
Title	图表	作为一个图表的标题进行显示的文本
To Value of	布局对象	在对象选择中一个参数被设置的值，该参数是在Set Parameter属性中说明的
Type	图表、参数	图表的基类，可取值有：Column、Pie、Bar、Table、Line、Scatter、Mixed、High-Low、Double-Y、Gantt。对一个参数，数据类型有：Number、Char、Date
Type	查询	查询的类型，可取值有：SQL语句、SYLK文件(电子表格)、WKS文件(Lotus 1-2-3电子表格)、PRN文件(文本文件)、外部SQL文件(文件中的一条SELECT语句)。文件的名字出现在File属性中
Update Chart on Query Execution	图表	当用户执行查询时，确定是否要自动地对图表进行更新，缺省值是选中的
Volume	声音	录制或回放该声音的音量

China-pub.com
下载

第六部分　附　　录

附录A　数　据　库

本附录包括了Talbot数据库完整的Oracle 8 SQL定义。参看在第3章对不同表的说明，和完整的用图解数据库模型显示的表之间的关系。

用户可以通过使用CONNECT和RESOURCE权限注册为单一Oracle用户来使用这些命令。尽管这不是一个大数据库，如果想更好地控制数据的大小，可以为表添加工作空间约束条件。

A.1　删除表

第一组命令删除表、序列和类型，破坏数据库中所有的数据。这些命令确保由空白的示例数据库开始。

```
DROP TABLE WorkerHasSkill;
DROP TABLE Skill;
DROP TABLE Ledger;
DROP TABLE Person;
DROP TABLE Lodging;
DROP SEQUENCE LedgerSequence;
DROP TYPE WorkerHasSkill_TP;
DROP TYPE Skill_TP;
DROP TYPE Ledger_TP;
DROP TYPE Person_TP;
DROP TYPE Lodging_TP;
DROP TYPE Address_TP;
```

A.2　创建物理和逻辑模式

下面命令创建存储结构和模式(物理数据库和概念数据库)。键入这些命令时，不要忘记在CREATE TYPE语句后加上斜线(/)。因为其中包含PL/SQL，需要斜线和分号结束符来执行语句。

```
CREATE SEQUENCE LedgerSequence;
CREATE TYPE Address_TP AS OBJECT (
  Street1    VARCHAR2(100), -- first part of street address
  Street2    VARCHAR2(100), -- second part of street address
  City       VARCHAR2(100), -- city name
  State      VARCHAR2(2),   -- two-letter state abbreviation
  PostalCode VARCHAR2(15),  -- postal code of address
  Country    VARCHAR2(50)   -- country
);
/
CREATE TYPE Lodging_TP AS OBJECT (
  Lodging       VARCHAR2(15), -- short name for the lodging
  LongName      VARCHAR2(40), -- complete name
  Manager       VARCHAR2(25), -- manager's name
  Address       Address_TP    -- address of the lodging
);
/
```

China-pub.com 下载

```
CREATE TYPE Person_TP AS OBJECT (
  Name          VARCHAR2(25),     -- worker's name
  AgeNUMBER,          -- age in years
  Lodging       REF Lodging_TP  -- reference to person's lodging
);
/
CREATE TYPE Ledger_TP AS OBJECT (
  LedgerID       NUMBER,        -- sequence number, primary key
  ActionDate     DATE,           -- when
  Action         VARCHAR2(8),    -- bought, sold, paid, received
  ItemVARCHAR2(30),  -- what
  Quantity       NUMBER,        -- how many
  QuantityType   VARCHAR2(10),  -- type of quantity: lbs, bushels, etc.
  RateNUMBER(9,2),    -- how much per quantity type
  Amount         NUMBER(9,2),    -- total amount (extension) rate * quantity
  Person         REF Person_TP  -- reference to subject of transaction
);
/
CREATE TYPE Skill_TP AS OBJECT (
  Skill        VARCHAR2(25), -- name of a capability
  Description  VARCHAR2(80)  -- description of the skill
);
/
REM Note that you cannot have a REF attribute as part of the primary key
CREATE TYPE WorkerHasSkill_TP AS OBJECT (
  Name          VARCHAR2(25),  -- worker's name, part of key
  Skill         VARCHAR2(25),  -- capability name, part of key
  Ability       VARCHAR2(15)   -- how skilled is the worker?
);
/
CREATE TABLE Ledger OF Ledger_TP (
  PRIMARY KEY (LedgerID)
);
CREATE TABLE Lodging OF Lodging_TP (
  PRIMARY KEY (Lodging)
);
CREATE TABLE Skill OF Skill_TP (
  PRIMARY KEY (Skill)
);
CREATE TABLE Person OF Person_TP (
  PRIMARY KEY (Name)
);
CREATE TABLE WorkerHasSkill OF WorkerHasSkill_TP (
  PRIMARY KEY (Name, Skill),
  FOREIGN KEY (Name) REFERENCES Person(Name),
  FOREIGN KEY (Skill) REFERENCES Skill(Skill)
);
```

A.3 创建数据

以下INSERT语句创建实际数据库。必须按下面的顺序加载表以满足所有的完整性约束。例如，如果插入包含Lodging的Person，而在此之前没有插入Lodging，则违反了一个完整性约束。

A.3.1 Lodging表

```
INSERT INTO Lodging VALUES (
'Cranmer','Cranmer Retreat House','Thom Cranmer',
Address_TP('Hill St.', NULL, 'Berkeley' , 'NH', '03431', 'USA'));
INSERT INTO Lodging VALUES (
```

China-pub.com
下载


```
'Matts','Matts Long Bunk House','Roland Brandt',
Address_TP('3 Mile Rd.', NULL, 'Keene', 'NH', '03431', 'USA'));
INSERT INTO Lodging VALUES (
'Mullers','Mullers Coed Lodging','Ken Muller',
Address_TP('120 Main', NULL, 'Edmeston', 'NH', '03431', 'USA'));
INSERT INTO Lodging VALUES (
'Papa King','Papa King Rooming','William King',
Address_TP('127 Main', NULL, 'Edmeston', 'NH', '03431', 'USA'));
INSERT INTO Lodging VALUES (
'Rose Hill','Rose Hill for Men','John Peletier',
Address_TP('RFD 3', NULL, 'N. Edmeston', 'NH', '03431', 'USA'));
INSERT INTO Lodging VALUES (
'Weitbrocht','Weitbrocht Rooming','Eunice Benson',
Address_TP('320 Geneva', NULL, 'Keene', 'NH', '03431', 'USA'));
```

A.3.2 Skill表

```
INSERT INTO Skill VALUES ('Woodcutter',
'Mark And Fell Trees, Split, Stack, Haul');
INSERT INTO Skill VALUES ('Combine Driver',
'Harness, Drive, Groom Horses, Adjust Blades');
INSERT INTO Skill VALUES ('Smithy',
'Stack For Fire, Run Bellows, Cut, Shoe Horses');
INSERT INTO Skill VALUES ('Grave Digger',
'Mark And Cut Sod, Dig, Shore, Fill, Resod');
INSERT INTO Skill VALUES ('Discus',
'Harness, Drive, Groom Horses, Blade Depth');
INSERT INTO Skill VALUES ('Work','General Unskilled labor');
```

A.3.3 Person表

```
INSERT INTO Person
  SELECT 'Bart Sarjeant', 22, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Cranmer';
INSERT INTO Person
  SELECT 'Elbert Talbot', 43, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Weitbrocht';
INSERT INTO Person
  SELECT 'Donald Rollo', 16, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Matts';
INSERT INTO Person
  SELECT 'Jed Hopkins', 33, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Matts';
INSERT INTO Person
  SELECT 'William Swing', 15, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Cranmer';
INSERT INTO Person
  SELECT 'John Pearson', 27, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Rose Hill';
INSERT INTO Person
```

China-pub.com 下载

```
  SELECT 'George Oscar', 41, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Rose Hill';
INSERT INTO Person
  SELECT 'Kay And Palmer Wallbom', NULL, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Rose Hill';
INSERT INTO Person
  SELECT 'Pat Lavay', 21, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Rose Hill';
INSERT INTO Person
  SELECT 'Richard Koch And Brothers', NULL, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Weitbrocht';
INSERT INTO Person
  SELECT 'Dick Jones', 18, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Rose Hill';
INSERT INTO Person
  SELECT 'Adah Talbot', 23, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Papa King';
INSERT INTO Person
  SELECT 'Roland Brandt', 35, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Matts';
INSERT INTO Person
  SELECT 'Peter Lawson', 25, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Cranmer';
INSERT INTO Person
  SELECT 'Victoria Lynn', 32, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Mullers';
INSERT INTO Person VALUES ('Wilfred Lowell', 67, NULL);
INSERT INTO Person VALUES ('Helen Brandt', 15, NULL);
INSERT INTO Person
  SELECT 'Gerhardt Kentgen', 55, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Papa King';
INSERT INTO Person
  SELECT 'Andrew Dye', 29, REF(L)
    FROM Lodging L
   WHERE Lodging = 'Rose Hill';
INSERT INTO Person VALUES ('Blacksmith', NULL, NULL);
INSERT INTO Person VALUES ('Boole And Jones', NULL, NULL);
INSERT INTO Person VALUES ('Dean Foreman', NULL, NULL);
INSERT INTO Person VALUES ('Dr. Carlstrom', NULL, NULL);
INSERT INTO Person VALUES ('Edward Johnson', NULL, NULL);
INSERT INTO Person VALUES ('Edythe Gammiere', NULL, NULL);
INSERT INTO Person VALUES ('Feed Store', NULL, NULL);
INSERT INTO Person VALUES ('Fred Fuller', NULL, NULL);
INSERT INTO Person VALUES ('Gary Kentgen', NULL, NULL);
INSERT INTO Person VALUES ('General Store', NULL, NULL);
INSERT INTO Person VALUES ('George August', NULL, NULL);
INSERT INTO Person VALUES ('George B. McCormick', NULL, NULL);
```

China-pub.com
下载


```
INSERT INTO Person VALUES ('Harold Schole', NULL, NULL);
INSERT INTO Person VALUES ('Henry Chase', NULL, NULL);
INSERT INTO Person VALUES ('Isaiah James', NULL, NULL);
INSERT INTO Person VALUES ('James Cole', NULL, NULL);
INSERT INTO Person VALUES ('Janice Talbot', NULL, NULL);
INSERT INTO Person VALUES ('John Austin', NULL, NULL);
INSERT INTO Person VALUES ('Lily Carlstrom', NULL, NULL);
INSERT INTO Person VALUES ('Livery', NULL, NULL);
INSERT INTO Person VALUES ('Mill', NULL, NULL);
INSERT INTO Person VALUES ('Manner Jewelers', NULL, NULL);
INSERT INTO Person VALUES ('Methodist Church', NULL, NULL);
INSERT INTO Person VALUES ('Morris Arnold', NULL, NULL);
INSERT INTO Person VALUES ('Palmer Wallbom', NULL, NULL);
INSERT INTO Person VALUES ('Phone Company', NULL, NULL);
INSERT INTO Person VALUES ('Post Office', NULL, NULL);
INSERT INTO Person VALUES ('Quarry', NULL, NULL);
INSERT INTO Person VALUES ('Robert James', NULL, NULL);
INSERT INTO Person VALUES ('Sam Dye', NULL, NULL);
INSERT INTO Person VALUES ('School', NULL, NULL);
INSERT INTO Person VALUES ('Underwood Bros', NULL, NULL);
INSERT INTO Person VALUES ('Verna Hardware', NULL, NULL);
```

A.3.4 WorkerHasSkill表

```
INSERT INTO WorkerHasSkill
VALUES ('Dick Jones','Smithy','Excellent');
INSERT INTO WorkerHasSkill
VALUES ('John Pearson','Combine Driver',NULL);
INSERT INTO WorkerHasSkill
VALUES ('John Pearson','Smithy','Average');
INSERT INTO WorkerHasSkill
VALUES ('Helen Brandt','Combine Driver','Very Fast');
INSERT INTO WorkerHasSkill
VALUES ('John Pearson','Woodcutter','Good');
INSERT INTO WorkerHasSkill
VALUES ('Victoria Lynn','Smithy','Precise');
INSERT INTO WorkerHasSkill
VALUES ('Adah Talbot','Work','Good');
INSERT INTO WorkerHasSkill
VALUES ('Wilfred Lowell','Work','Average');
INSERT INTO WorkerHasSkill
VALUES ('Elbert Talbot','Discus','Slow');
INSERT INTO WorkerHasSkill
VALUES ('Wilfred Lowell','Discus','Average');
```

A.3.5 Ledger表

```
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'01-APR-01','Paid','Plowing',1,
       'Day',3,3,REF(P)
  FROM Person P
 WHERE Name = 'Richard Koch And Brothers';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'02-MAY-01','Paid','Work',1,
       'Day',1,1,REF(P)
  FROM Person P
```

China-pub.com 下载

```
 WHERE Name = 'Dick Jones';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'03-JUN-01','Paid','Work',1,
       'Day',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Elbert Talbot';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'04-JAN-01','Paid','Work',1,
       'Day',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Gerhardt Kentgen';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'04-FEB-01','Paid','Work',.5,
       'Day',1,.5, REF(P)
  FROM Person P
 WHERE Name = 'Elbert Talbot';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'05-APR-01','Paid','Work',1,
       'Day',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Dick Jones';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'06-AUG-01','Paid','Plowing',1,
       'Day',1.8,1.8, REF(P)
  FROM Person P
 WHERE Name = 'Victoria Lynn';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'07-OCT-01','Paid','Plowing',.5,
       'Day',3,1.5, REF(P)
  FROM Person P
 WHERE Name = 'Richard Koch And Brothers';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'09-SEP-01','Paid','Work',1,
       'Day',1,1,REF(P)
  FROM Person P
 WHERE Name = 'Adah Talbot';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'09-OCT-01','Paid','Work',.5,
       'Day',1.25,.63, REF(P)
  FROM Person P
 WHERE Name = 'Donald Rollo';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'10-NOV-01','Paid','Work',1,
       'Day',1.25,.63, REF(P)
  FROM Person P
 WHERE Name = 'John Pearson';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'10-AUG-01','Paid','Work',1,
       'Day',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Helen Brandt';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'11-AUG-01','Paid','Work',1,
       'Day',2,2, REF(P)
  FROM Person P
 WHERE Name = 'Helen Brandt';
INSERT INTO Ledger
```

China-pub.com
下载


```
SELECT LedgerSequence.NEXTVAL,'11-SEP-01','Paid','Work',1,
       'Day',.75,.75, REF(P)
  FROM Person P
 WHERE Name = 'Roland Brandt';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'12-DEC-01','Paid','Work',1,
       'Day',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Bart Sarjeant';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'12-JAN-01','Paid','Work',1,
       'Day',1,1, REF(P)
  FROM Person P
 WHERE Name = 'George Oscar';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'13-JUN-01','Paid','Work',1,
       'Day',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Peter Lawson';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'14-JUL-01','Paid','Work',1,
       'Day',1.2,1.2, REF(P)
  FROM Person P
 WHERE Name = 'Wilfred Lowell';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'15-JUL-01','Paid','Work',1,
       'Day',2.25,2.25, REF(P)
  FROM Person P
 WHERE Name = 'Kay And Palmer Wallbom';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'03-OCT-01','Sold','Boot Between Horses',1,
       'Each',12.5,12.5, REF(P)
  FROM Person P
 WHERE Name = 'Gary Kentgen';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'01-NOV-01','Bought','Calf',2,
       'Each',2,4, REF(P)
  FROM Person P
 WHERE Name = 'Gary Kentgen';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'02-NOV-01','Bought','Mare',1,
       'Each',5,5, REF(P)
  FROM Person P
 WHERE Name = 'James Cole';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'03-NOV-01','Bought','Pig',1,
       'Each',2,2, REF(P)
  FROM Person P
 WHERE Name = 'Andrew Dye';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'04-NOV-01','Bought','Hay',1,
       'Wagon',5,5, REF(P)
  FROM Person P
 WHERE Name = 'Andrew Dye';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'05-NOV-01','Bought','Hay',4,
       'Wagon',5,20, REF(P)
```

China-pub.com 下载

```
  FROM Person P
 WHERE Name = 'Andrew Dye';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'05-NOV-01','Bought','Line',1,
       'Set',.75,.75, REF(P)
  FROM Person P
 WHERE Name = 'Andrew Dye';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'06-NOV-01','Bought','Colt',2,
       'Each',4.5,9, REF(P)
  FROM Person P
 WHERE Name = 'Andrew Dye';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'06-AUG-01','Paid','Plowing',2,
       'Day',2,4, REF(P)
  FROM Person P
 WHERE Name = 'Andrew Dye';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'07-NOV-01','Paid','Sawed Wood',1,
       'Day',.5,.5, REF(P)
  FROM Person P
 WHERE Name = 'Andrew Dye';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'09-NOV-01','Bought','Colt',1,
       'Each',10,10, REF(P)
  FROM Person P
 WHERE Name = 'Andrew Dye';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'10-NOV-01','Sold','Hefer',1,
       'Each',28,28, REF(P)
  FROM Person P
 WHERE Name = 'Pat Lavay';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'11-NOV-01','Sold','Boot Between Horses',1,
       'Each',6,6, REF(P)
  FROM Person P
 WHERE Name = 'Pat Lavay';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'11-NOV-01','Sold','Butter',1,
       'Pound',.15,.15, REF(P)
  FROM Person P
 WHERE Name = 'Pat Lavay';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'12-NOV-01','Paid','Work',2,
       'Day',.75,1.5, REF(P)
  FROM Person P
 WHERE Name = 'Pat Lavay';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'13-NOV-01','Paid','Cut Logs',.5,
       'Day',.5,.25, REF(P)
  FROM Person P
 WHERE Name = 'Pat Lavay';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'13-NOV-01','Paid','Drawed Logs',1.5,
       'Day',.5,.75, REF(P)
  FROM Person P
 WHERE Name = 'Pat Lavay';
```

China-pub.com
下载


```
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'13-DEC-01','Paid','Sawed Wood',1,
       'Day',.5,.5, REF(P)
  FROM Person P
 WHERE Name = 'Pat Lavay';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'14-NOV-01','Sold','Hefer',1,
       'Each',35,35, REF(P)
  FROM Person P
 WHERE Name = 'Morris Arnold';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'15-NOV-01','Sold','Beef',37,
       'Pound',.04,1.48, REF(P)
  FROM Person P
 WHERE Name = 'Fred Fuller';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'16-NOV-01','Sold','Butter',5,
       'Pound',.16,.8, REF(P)
  FROM Person P
 WHERE Name = 'Victoria Lynn';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'18-NOV-01','Sold','Butter',6,
       'Pound',.16,.96, REF(P)
  FROM Person P
 WHERE Name = 'John Pearson';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'20-NOV-01','Sold','Heifer',1,
       'Each',30,30, REF(P)
  FROM Person P
 WHERE Name = 'Palmer Wallbom';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'21-NOV-01','Sold','Beef',116,
       'Pound',.06,6.96, REF(P)
  FROM Person P
 WHERE Name = 'Roland Brandt';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'22-NOV-01','Sold','Beef',118,
       'Pound',.06,7.08, REF(P)
  FROM Person P
 WHERE Name = 'Gerhardt Kentgen';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'01-DEC-01','Bought','Beef',138,
       'Pound',.05,6.9, REF(P)
  FROM Person P
 WHERE Name = 'Victoria Lynn';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'01-DEC-01','Bought','Beef',130,
       'Pound',.06,7.8, REF(P)
  FROM Person P
 WHERE Name = 'George B. McCormick';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'03-DEC-01','Bought','Beef',130,
       'Pound',.05,6.5, REF(P)
  FROM Person P
 WHERE Name = 'Peter Lawson';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'03-DEC-01','Bought','Beef',125,
```

China-pub.com 下载

```
        'Pound',.06,7.5, REF(P)
  FROM Person P
 WHERE Name = 'Helen Brandt';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'05-DEC-01','Bought','Beef',140,
        'Pound',.05,7, REF(P)
  FROM Person P
 WHERE Name = 'Robert James';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'05-DEC-01','Bought','Beef',145,
        'Pound',.05,7.25, REF(P)
  FROM Person P
 WHERE Name = 'Isaiah James';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'07-DEC-01','Bought','Horse',1,
        'Each',30,30, REF(P)
  FROM Person P
 WHERE Name = 'George August';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'07-DEC-01','Bought','Reaper/Binder',1,
        'Each',47.5,47.5, REF(P)
  FROM Person P
 WHERE Name = 'Janice Talbot';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'03-JAN-01','Bought','Hominy',1,
        'Bushel',1.25,1.25, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'09-JAN-01','Bought','Lice Killer',1,
        'Each',.5,.5, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'11-JAN-01','Bought','Mending Brace',1,
        'Each',.15,.15, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'11-JAN-01','Bought','Stove Blacking',1,
        'Each',.05,.05, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'13-JAN-01','Bought','Grinding Bat',10,
        'Each',.03,.3, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'14-JAN-01','Sold','Beef Hide',1,
        'Each',5.46,5.46, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'14-JAN-01','Sold','Cheese Flat',13,
        'Each',3.15,40.95, REF(P)
  FROM Person P
```

China-pub.com
下载


```
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'14-JAN-01','Bought','Lantern Globe',1,
       'Each',.1,.1, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'15-JAN-01','Bought','Stamp For Letter',1,
       'Each',.02,.02, REF(P)
  FROM Person P
 WHERE Name = 'Post Office';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'15-JAN-01','Bought','Stocking',2,
       'Pair',.15,.3, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'16-JAN-01','Bought','Oil',4,
       'Gallon',.1,.4, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'16-JAN-01','Bought','Sugar',25,
       'Pound',.07,1.75, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'16-JAN-01','Bought','Molasses',1,
       'Gallon',.6,.6, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'16-JAN-01','Bought','Card Of Thanks',1,
       'Each',.3,.3, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'17-JAN-01','Bought','Horse Shodding',1,
       'Each',.85,.85, REF(P)
  FROM Person P
 WHERE Name = 'Livery';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'17-JAN-01','Bought','Corn',230,
       'Pound',.01,2.3, REF(P)
  FROM Person P
 WHERE Name = 'Feed Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'18-JAN-01','Bought','Corn Meal',213,
       'Pound',.01,2.13, REF(P)
  FROM Person P
 WHERE Name = 'Feed Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'18-JAN-01','Bought','Paper',50,
       'Sheets',.01,.5, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
```

China-pub.com 下载

```
SELECT LedgerSequence.NEXTVAL,'18-JAN-01','Bought','Coffee',1,
       'Pound',.3,.3, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'18-JAN-01','Bought','Seeded Raisins',1,
       'Pound',.12,.12, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'18-JAN-01','Bought','Cotton Stocking',3,
       'Pair',.08,.24, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'19-JAN-01','Bought','Cotton Stocking',3,
       'Pair',.08,.24, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'19-JAN-01','Bought','Grinding Bat',24,
       'Each',.03,.72, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'19-JAN-01','Bought','Telephone Call',1,
       'Each',.15,.15, REF(P)
  FROM Person P
 WHERE Name = 'Phone Company';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'19-JAN-01','Bought','Tea',.5,
       'Pound',.5,.25, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'19-JAN-01','Bought','Hat',1,
       'Each',.1,.1, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'19-JAN-01','Bought','Salt Peter',1,
       'Each',.08,.08, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'19-JAN-01','Bought','Envelopes',6,
       'Each',.02,.12, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'19-JAN-01','Bought','Creoal',2,
       'Qaurt',.37,.74, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'23-JAN-01','Sold','Wood',1,
       'Cord',2,2, REF(P)
```

China-pub.com
下载


```
  FROM Person P
 WHERE Name = 'Methodist Church';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'24-JAN-01','Bought','Schooling',1,
       'Each',1,1, REF(P)
  FROM Person P
 WHERE Name = 'School';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'24-JAN-01','Bought','Hominy',186,
       'Each',.01,1.86, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'28-JAN-01','Bought','Grinding',1,
       'Each',.9,.9, REF(P)
  FROM Person P
 WHERE Name = 'Mill';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'28-JAN-01','Bought','Popcorn',5,
       'Pound',.04,.2, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'02-FEB-01','Bought','Sulpher',5,
       'Pound',.25,1.25, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'03-FEB-01','Bought','Oil',4,
       'Gallon',.13,.52, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'03-FEB-01','Bought','Swamp Root',1,
       'Each',.75,.75, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'04-FEB-01','Bought','Shoeing Ned',1,
       'Each',.5,.5, REF(P)
  FROM Person P
 WHERE Name = 'Blacksmith';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'04-FEB-01','Bought','Grinding',1,
       'Each',.47,.47, REF(P)
  FROM Person P
 WHERE Name = 'Mill';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'05-FEB-01','Bought','Pills',1,
       'Each',.25,.25, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'07-FEB-01','Bought','Thread',2,
       'Each',.05,.1, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
```

China-pub.com 下载

```
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'08-FEB-01','Bought','Shirts',2,
       'Each',.5,1, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'10-FEB-01','Sold','Butter',9,
       'Pound',.25,2.25, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'18-FEB-01','Bought','Horse Medison',1,
       'Each',.13,.13, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'18-FEB-01','Bought','Elbo Stove Pipe',1,
       'Each',.15,.15, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'18-FEB-01','Sold','Calf',1,
       'Each',4,4, REF(P)
  FROM Person P
 WHERE Name = 'Lily Carlstrom';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'25-FEB-01','Sold','Butter',21,
       'Pound',.25,5.25, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'28-FEB-01','Bought','Swamp Root',1,
       'Each',.75,.75, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'28-FEB-01','Bought','Liver Pills',1,
       'Each',.2,.2, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'28-FEB-01','Sold','Butter',3,
       'Pound',.25,.75, REF(P)
  FROM Person P
 WHERE Name = 'Helen Brandt';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'01-APR-01','Bought','Grinding',1,
       'Each',.45,.45, REF(P)
  FROM Person P
 WHERE Name = 'Mill';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'06-MAR-01','Bought','Medison For
       Indigestion',1,
       'Each',.4,.4, REF(P)
  FROM Person P
 WHERE Name = 'Dr. Carlstrom';
INSERT INTO Ledger
```

China-pub.com
下载


```
SELECT LedgerSequence.NEXTVAL,'06-JUN-01','Bought','Breading Powder',1,
       'Each',.9,.9, REF(P)
  FROM Person P
 WHERE Name = 'Mill';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'06-MAR-01','Bought','Pants',1,
       'Pair',.75,.75, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'07-APR-01','Bought','Hominy',200,
       'Pound',.01,2, REF(P)
  FROM Person P
 WHERE Name = 'Mill';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'08-MAR-01','Bought','Tobacco For
       Lice',1,
       'Each',.25,.25, REF(P)
  FROM Person P
 WHERE Name = 'Mill';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'07-MAR-01','Bought','Shoeing',1,
       'Each',.35,.35, REF(P)
  FROM Person P
 WHERE Name = 'Blacksmith';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'07-APR-01','Bought','Pins',1,
       'Each',.05,.05, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'07-MAR-01','Bought','Mail Box',1,
       'Each',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Post Office';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'10-MAR-01','Bought','Stove Pipe Thimbles',2,
       'Each',.5,1, REF(P)
  FROM Person P
 WHERE Name = 'Verna Hardware';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'13-MAR-01','Bought','Thermometer',1,
       'Each',.15,.15, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'14-MAR-01','Bought','Lot In Cemetery No. 80',1,
       'Each',25,25,REF(P)
  FROM Person P
 WHERE Name = 'Methodist Church';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'14-MAR-01','Paid','Digging Of Grave',1,
       'Each',3,3, REF(P)
  FROM Person P
 WHERE Name = 'Jed Hopkins';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'16-APR-01','Bought','Grinding',1,
```

China-pub.com 下载

```
        'Each',.16,.16, REF(P)
  FROM Person P
 WHERE Name = 'Mill';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'16-MAR-01','Bought','Grinding',1,
        'Each',.16,.16, REF(P)
  FROM Person P
 WHERE Name = 'Mill';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'23-MAR-01','Bought','Cloth For Dress Lining',2,
        'Yard',.27,.54,REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'18-AUG-01','Bought','SYRUP Thermometer',1,
        'Each',1,1, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'25-MAR-01','Bought','Boots For Shirley',1,
        'Pair',2.5,2.5, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'27-APR-01','Bought','Syrup Cans',2,
        'Dozen',1.07,2.14, REF(P)
  FROM Person P
 WHERE Name = 'Verna Hardware';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'22-MAR-01','Bought','Milk Cans',2,
        'Each',2.5,5, REF(P)
  FROM Person P
 WHERE Name = 'Verna Hardware';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'23-APR-01','Bought','Dubble Strainer',1,
        'Each',.95,.95, REF(P)
  FROM Person P
 WHERE Name = 'Verna Hardware';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'25-JUN-01','Bought','Milk Stirrer',1,
        'Each',.25,.25, REF(P)
  FROM Person P
 WHERE Name = 'Verna Hardware';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'27-MAR-01','Bought','Hominy',77,
        'Pound',.01,.77, REF(P)
  FROM Person P
 WHERE Name = 'Mill';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'28-APR-01','Bought','Corn',104,
        'Pound',.01,1.04, REF(P)
  FROM Person P
 WHERE Name = 'Mill';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'06-APR-01','Bought','Funeral',1,
        'Each',3.19,3.19, REF(P)
  FROM Person P
```

China-pub.com
下载


```
 WHERE Name = 'Underwood Bros';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'30-APR-01','Bought','Brush',1,
       'Each',.05,.05, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'30-APR-01','Bought','Sand',5,
       'Bushel',.03,.15, REF(P)
  FROM Person P
 WHERE Name = 'Quarry';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'31-MAR-01','Sold','Molasses',3,
       'Gallon',1,3, REF(P)
  FROM Person P
 WHERE Name = 'Harold Schole';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'28-MAR-01','Sold','Molasses',1,
       'Gallon',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Gerhardt Kentgen';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'30-MAR-01','Bought','Fixing Shirleys Watch',1,
       'Each',.25,.25,REF(P)
  FROM Person P
 WHERE Name = 'Manner Jewelers';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'04-APR-01','Sold','Butter',9,
       'Pound',.23,2.07, REF(P)
  FROM Person P
 WHERE Name = 'Harold Schole';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'05-APR-01','Bought','Soda',1,
       'Each',.05,.05, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'05-MAR-01','Bought','Telephone Call',1,
       'Each',.2,.2, REF(P)
  FROM Person P
 WHERE Name = 'Phone Company';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'06-APR-01','Bought','Gloves',1,
       'Pair',.25,.25, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'06-APR-01','Bought','Shoes For Shirley',1,
       'Pair',2,2, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'09-APR-01','Bought','Peanuts',1,
       'Each',.05,.05, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
```

China-pub.com 下载

```
SELECT LedgerSequence.NEXTVAL,'11-APR-01','Bought','Bran',300,
       'Pound',.01,3, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'15-APR-01','Bought','Shoeing',2,
       'Each',.3,.6, REF(P)
  FROM Person P
 WHERE Name = 'Blacksmith';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'17-APR-01','Bought','Hominy',173,
       'Pound',.01,1.73, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'17-APR-01','Bought','Bran',450,
       'Pound',.01,4.5, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'17-APR-01','Bought','Calf Meal',110,
       'Pound',.01,1.1, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'22-APR-01','Bought','Hominy',454,
       'Pound',.01,4.54, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'22-APR-01','Bought','Bran',300,
       'Pound',.01,3, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'22-APR-01','Sold','Calf',1,
       'Each',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Pat Lavay';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'25-APR-01','Bought','Calf Meal',100,
       'Each',.01,1, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'27-APR-01','Bought','Shoeing Ned',1,
       'Each',.5,.5, REF(P)
  FROM Person P
 WHERE Name = 'Blacksmith';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'07-JUN-01','Received','Breaking Colt',1,
       'Each',5,5, REF(P)
  FROM Person P
 WHERE Name = 'Sam Dye';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'07-JUN-01','Received','Keeping Colt',1,
       'Each',4,4, REF(P)
```

China-pub.com
下载


```
  FROM Person P
 WHERE Name = 'Sam Dye';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'17-JUN-01','Bought','School Tax',1,
       'Each',6.56,6.56, REF(P)
  FROM Person P
 WHERE Name = 'School';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'17-JUN-01','Received','Threshing',2,
       'Day',1,2, REF(P)
  FROM Person P
 WHERE Name = 'Henry Chase';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'18-JUN-01','Paid','Threshing',.5,
       'Day',1,.5, REF(P)
  FROM Person P
 WHERE Name = 'William Swing';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'18-JUN-01','Bought','Sheep',22,
       'Each',.87,19.14, REF(P)
  FROM Person P
 WHERE Name = 'Boole And Jones';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'15-MAR-01','Sold','Potatoes',5,
       'Bushel',.25,1.25, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'15-MAR-01','Sold','Cow',2,
       'Each',33,66, REF(P)
  FROM Person P
 WHERE Name = 'Sam Dye';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'15-MAR-01','Received','Boot Between Horses',1,
       'Each',10,10, REF(P)
  FROM Person P
 WHERE Name = 'Adah Talbot';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'18-MAR-01','Sold','Wagon',1,
       'Each',5,5, REF(P)
  FROM Person P
 WHERE Name = 'Adah Talbot';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'04-APR-01','Sold','Harnes',1,
       'Each',2,2, REF(P)
  FROM Person P
 WHERE Name = 'Adah Talbot';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'16-APR-01','Sold','Cow',3,
       'Each',30,90, REF(P)
  FROM Person P
 WHERE Name = 'George B. McCormick';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'09-JUN-01','Bought','Use Of Pasture',1,
       'Each',10,10, REF(P)
  FROM Person P
 WHERE Name = 'George B. McCormick';
INSERT INTO Ledger
```

China-pub.com 下载

```
SELECT LedgerSequence.NEXTVAL,'28-JUN-01','Bought','Sheep And Bull',1,
       'Lot',97.88,97.88, REF(P)
  FROM Person P
 WHERE Name = 'Edward Johnson';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'03-JUL-01','Sold','Heifer',1,
       'Each',35,35, REF(P)
  FROM Person P
 WHERE Name = 'Sam Dye';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'18-MAY-01','Bought','Middlings',180,
       'Pound',.01,1.8, REF(P)
  FROM Person P
 WHERE Name = 'Dean Foreman';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'20-MAY-01','Bought','Middlings',450,
       'Pound',.01,4.5, REF(P)
  FROM Person P
 WHERE Name = 'George Oscar';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'22-MAY-01','Bought','Middlings',640,
       'Pound',.01,6.4, REF(P)
  FROM Person P
 WHERE Name = 'Edythe Gammiere';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'23-MAY-01','Bought','Middlings',110,
       'Pound',.01,1.1, REF(P)
  FROM Person P
 WHERE Name = 'John Austin';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'28-MAY-01','Bought','Comb',1,
       'Each',.07,.07, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'29-MAY-01','Bought','Buttons',1,
       'Each',.1,.1, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'05-JUL-01','Bought','Beans',6,
       'Pound',.03,.18, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'29-MAY-01','Bought','Raisons',3,
       'Pound',.08,.24, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'29-MAY-01','Bought','Cheese',3,
       'Pound',.09,.27, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'04-JUN-01','Bought','Beer',1,
       'Each',.2,.2, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
```

China-pub.com
下载


```
SELECT LedgerSequence.NEXTVAL,'04-JUN-01','Bought','Cough Syrup',1,
       'Each',.25,.25, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'26-JUN-01','Bought','Shoe String',2,
       'Pair',.04,.08, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'26-JUN-01','Bought','Close Pins',1,
       'Each',.05,.05, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'26-JUN-01','Bought','Close Brush',1,
       'Each',.1,.1, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'06-MAR-01','Sold','Eggs',14,
       'Dozen',.12,1.68, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'06-MAR-01','Sold','Hens',12,
       'Each',.5,6, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'15-APR-01','Sold','Eggs',13,
       'Dozen',.1,1.3, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'27-APR-01','Paid','Plowing',1,
       'Day',3,3, REF(P)
  FROM Person P
 WHERE Name = 'Richard Koch And Brothers';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'16-APR-01','Paid','Plowing',1,
       'Day',3,3, REF(P)
  FROM Person P
 WHERE Name = 'Richard Koch And Brothers';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'17-DEC-01','Paid','Sawing',1,
       'Day',.75,.75, REF(P)
  FROM Person P
 WHERE Name = 'Dick Jones';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'28-JUL-01','Paid','Sawing',1,
       'Day',.75,.75, REF(P)
  FROM Person P
 WHERE Name = 'Dick Jones';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'18-AUG-01','Paid','Weeding',1,
       'Day',.9,.9, REF(P)
  FROM Person P
 WHERE Name = 'Elbert Talbot';
INSERT INTO Ledger
```

China-pub.com 下载

```
SELECT LedgerSequence.NEXTVAL,'29-SEP-01','Paid','Work',1,
       'Day',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Gerhardt Kentgen';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'19-JAN-01','Paid','Work',1,
       'Day',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Gerhardt Kentgen';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'30-JAN-01','Paid','Work',.5,
       'Day',1,.5, REF(P)
  FROM Person P
 WHERE Name = 'Elbert Talbot';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'28-FEB-01','Paid','Work',1,
       'Day',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Elbert Talbot';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'20-MAR-01','Paid','Work',1,
       'Day',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Dick Jones';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'21-JUL-01','Paid','Work',1,
       'Day',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Victoria Lynn';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'22-OCT-01','Paid','Plowing',1,
       'Day',1.8,1.8, REF(P)
  FROM Person P
 WHERE Name = 'Dick Jones';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'23-SEP-01','Paid','Discus',.5,
       'Day',3,1.5, REF(P)
  FROM Person P
 WHERE Name = 'Richard Koch And Brothers';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'22-AUG-01','Paid','Sawing',1,
       'Day',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Peter Lawson';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'23-AUG-01','Paid','Sawing',1,
       'Day',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Peter Lawson';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'24-MAY-01','Paid','Work',1,
       'Day',1.2,1.2, REF(P)
  FROM Person P
 WHERE Name = 'Wilfred Lowell';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'11-MAY-01','Paid','Work',1,
       'Day',1.2,1.2, REF(P)
  FROM Person P
 WHERE Name = 'Wilfred Lowell';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'26-JUN-01','Paid','Painting',1,
```

China-pub.com
下载


```
        'Day',1.75,1.75, REF(P)
  FROM Person P
 WHERE Name = 'Kay And Palmer Wallbom';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'02-JUL-01','Bought','Middlings',220,
        'Pound',.01,2.2, REF(P)
  FROM Person P
 WHERE Name = 'Edythe Gammiere';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'03-JUL-01','Bought','Pig',1,
        'Each',3,3, REF(P)
  FROM Person P
 WHERE Name = 'John Austin';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'08-JUL-01','Bought','Cheese',1,
        'Pound',.09,.09, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'09-JUL-01','Bought','Beer',1,
        'Each',.2,.2, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'02-AUG-01','Bought','Milk Cans',3,
        'Each',2.5,7.5, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'05-AUG-01','Bought','Hominy',120,
        'Pound',.01,1.2, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'08-AUG-01','Bought','Brush',1,
        'Each',.06,.06, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'12-AUG-01','Bought','Corn',90,
        'Pound',.01,.9, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'25-MAR-01','Sold','Molasses',5,
        'Gallon',1,5, REF(P)
  FROM Person P
 WHERE Name = 'Sam Dye';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'29-AUG-01','Sold','Butter',5,
        'Pound',.23,1.15, REF(P)
  FROM Person P
 WHERE Name = 'Gerhardt Kentgen';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'06-SEP-01','Bought','Telephone Call',1,
        'Each',.2,.2, REF(P)
  FROM Person P
 WHERE Name = 'Phone Company';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'09-SEP-01','Bought','Peanuts',1,
        'Each',.05,.05, REF(P)
```

China-pub.com 下载

```
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'12-SEP-01','Bought','Bran',170,
       'Pound',.01,1.7, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'13-SEP-01','Bought','Shoeing',4,
       'Each',.3,1.2, REF(P)
  FROM Person P
 WHERE Name = 'Blacksmith';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'15-SEP-01','Bought','Hominy',144,
       'Pound',.01,1.44, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'20-APR-01','Bought','Bran',370,
       'Pound',.01,3.7, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'17-JUL-01','Bought','Calf Meal',90,
       'Pound',.01,.9, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'20-JUL-01','Bought','Hominy',300,
       'Pound',.01,3, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'25-JUL-01','Sold','Calf',1,
       'Each',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Sam Dye';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'19-SEP-01','Bought','Bran',100,
       'Pound',.01,1, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'23-SEP-01','Bought','Calf Meal',110,
       'Pound',.01,1.1, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'25-SEP-01','Bought','Hominy',80,
       'Pound',.01,.8, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'07-OCT-01','Paid','Work',1,
       'Day',1,1, REF(P)
  FROM Person P
 WHERE Name = 'Jed Hopkins';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'12-OCT-01','Bought','Sheep',12,
       'Each',.9,10.8, REF(P)
  FROM Person P
```

China-pub.com
下载


```
 WHERE Name = 'Boole And Jones';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'15-OCT-01','Sold','BEEF',935,'Pound',.03,28.05, REF(P)
  FROM Person P
 WHERE Name = 'General Store';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'18-OCT-01',
       'Received','Boot Between Horses',1,'Each',10,10, REF(P)
  FROM Person P
 WHERE Name = 'Adah Talbot';
INSERT INTO Ledger
SELECT LedgerSequence.NEXTVAL,'12-OCT-01','Sold','Heifer',1,
       'Each',35,35, REF(P)
  FROM Person P
 WHERE Name = 'George B. McCormick';
```

China-pub.com
下载

附录B　安装Oracle Developer

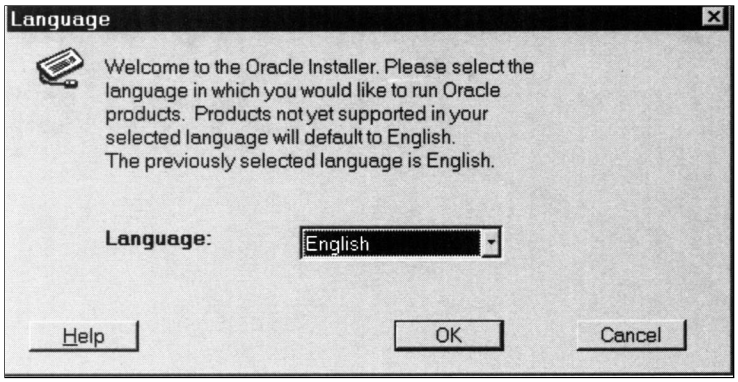
本书附带的CD-ROM包含了Oracle Developer版本6完整的使用版。本附录提供了CD-ROM中Developer(开发)软件和Deployment(展开)软件的基本安装指导。安装和配置Oracle Developer Server的细节参见第14章，只有在Windows NT和UNIX下Server Installation的安装选择项。本附录最后部分包含配置cartridges版本Oracle Developer Server(Forms、Reports和Graphics)的说明。

注意　安装软件后，应该阅读版本说明。在任务栏上选择Start｜Oracle Developer 6 Doc｜Release Note，即可看到版本说明。

B.1　安装Developer软件

使用Oracle Developer建立应用程序，需要从CD-ROM运行Development Installation。也可以选择运行Deployment Installation，细节参见B.2节“安装展开软件”。

1) 把CD-ROM插入要配置应用程序的系统的光盘驱动器中。如果系统的Autorun(自动执行)设置是打开的，则光盘自动执行，出现Language(语言)对话框，用户选择默认的语言。选择适当的语言后单击OK按钮(见图B-1)。



图B-1　Language对话框

如果系统的Autorun设置是关闭的，则打开浏览器找到CD-ROM根目录下的Setup程序，双击Setup程序开始Oracle Installer。

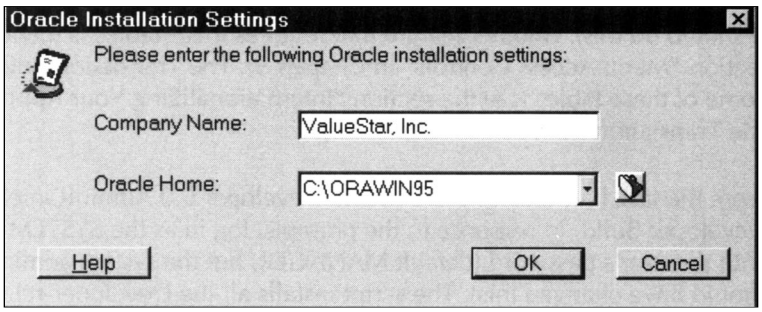
2) Oracle Installer显示Oracle Installation Setting(Oracle安装设置)对话框。输入要安装Oracle Developer的Oracle Home目录。单击OK(见图B-2)。

提示　尽管在UNIX下可以把Oracle Developer与其他Oracle软件安装到不同的目录下，但是在Windows下必须把Oracle Developer与其他Oracle软件安装到同一Oracle Home目录下。

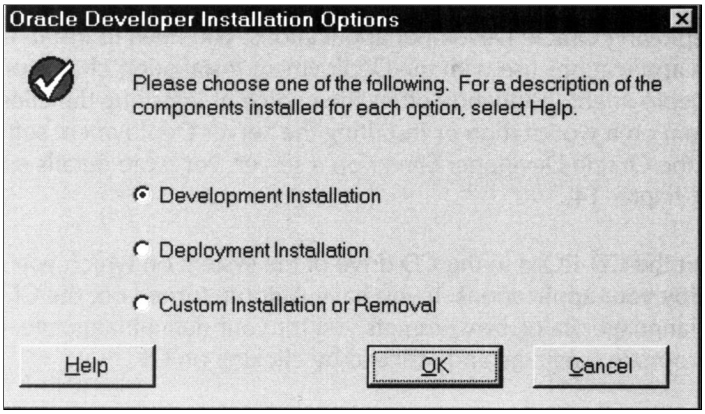
3) Oracle Installer显示Oracle Developer系统安装选择。接受默认的选择Development

China-pub.com
下载

Installation，然后单击OK(见图B-3)。



图B-2　Oracle Installation Settings对话框



图B-3　选择Development Installation选项

Oracle Installer现在为Developer环境安装所有的软件。根据用户的配置，可能会有几个问题要求用户确认，例如是否在 Microsoft Windows系统目录中安装文件 (为了OLE/ORA_FFI特性)。所有的这些确认问题都是不需加以说明的。下面安装表到 SYSTEM用户的步骤是可选的，除非需要使用它们的Oracle Developer特性。其中一个特性是保存应用或程序库到数据库 (参见第4章)。另外一个需要这些表的特性是菜单安全性 (参见第9章)。Translation Builder也需要这些表的一部分(参见第15章)。

1) 从任务栏选择Start|Oracle Developer 6.0 Admin|Oracle Developer Build。用SYSTEM用户和口令(默认为MANAGER，但系统管理员应该改变口令)登录。这个程序安装所有与Developer相关的表，这些表用来存储应用到数据库和使用菜单安全性。

2) 从任务栏选择Start| Oracle Developer 6.0 Admin|Oracle Developer Grant。用SYSTEM用户和口令登录，然后输入要授权访问的用户名。这个程序使用户能够访问 Oracle Developer的表。可以授权一个用户，使所有的开发者用这个用户登录，或者为每个开发者作单独的用户授权。

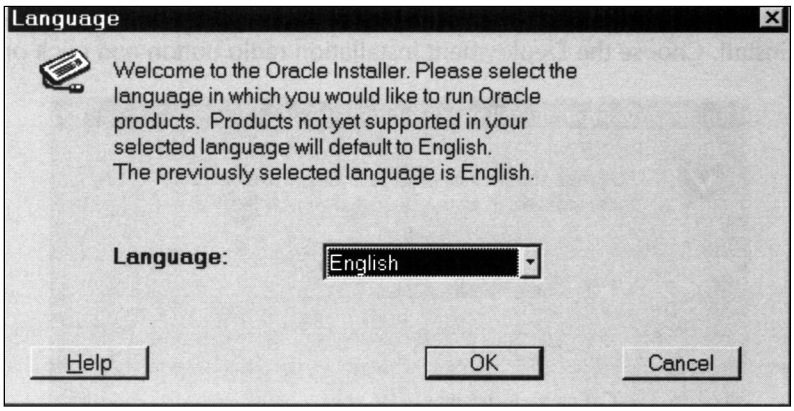
B.2　安装展开软件

如果要配置Oracle Developer应用程序，则需要用CD-ROM上的Deployment Installation选项安装应用程序需要使用的运行系统。 Deployment Installation提供了安装客户机/服务器运行

China-pub.com 下载

系统到工作站或安装使用户运行Oracle Developer Server的Server Deployment软件到服务器的选项。更多细节参见第14章。

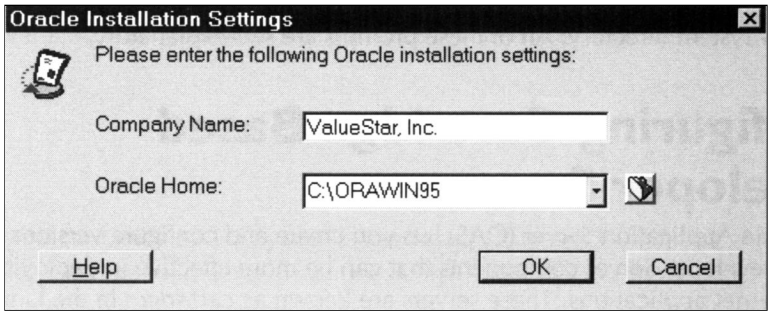
1) 把CD-ROM插入要配置应用程序的系统的光盘驱动器中。如果系统的 Autorun(自动执行)设置是打开的，则光盘自动执行，出现 Language对话框，用户选择默认的语言。选择适当的语言后单击OK按钮(见图B-4)。



图B-4 在Language对话框中选择默认的语言

如果系统的Autorun设置是关闭的，则打开浏览器找到CD-ROM根目录下的Setup程序，双击Setup程序开始Oracle Installer。

2) Oracle Installer显示Oracle Installation Settings(Oracle安装设置)对话框。输入要安装Oracle Developer的Oracle Home目录。单击OK(见图B-5)。



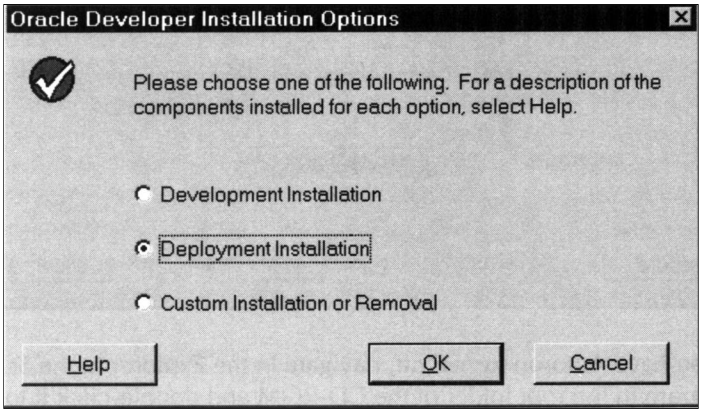
图B-5 Oracle Installation Settings对话框

提示 尽管在UNIX下可以把Oracle Developer与其他Oracle软件安装到不同的目录下，但是在Windows下必须把Oracle Developer与其他Oracle软件安装到同一Oracle Home目录下。

3) Oracle Installer显示Oracle Developer系统安装选择。接受默认的选择 Deployment Installation，然后单击OK(见图B-6)。

Oracle Installer现在为Developer运行系统配置安装所有的软件。安装所有的 Developer运行时可执行文件和一些附加组件，如 Query Builder，这些对用户可能会很有帮助。根据用户的配置，可能会有几个问题要求用户确认，例如是否在 Microsoft Windows系统目录中安装文件(为OLE/ORA_FFI特性)。所有的这些确认问题都是不需加以说明的。

China-pub.com
下载



图B-6　选择Deplogment Installation选项

B.3　配置Cartridge-Based版本Developer Servers

Oracle Application Server(OAS, Oracle应用服务器)使用户创建和配置各个版本的 Oracle Developer Servers组件，这些组件在配置和使用 Internet应用时是十分有效的。这些服务器在OAS中称作cartridge。这些cartridges动态地产生实现表单和报表需要 HTML页。Graphics Server cartridge为用户产生一个图形菜单在客户端浏览器显示。在第 14章介绍的静态HTML和CGI的交换技术表明动态服务器的建立是十分复杂的。这个部分介绍设置　Forms Server、Reports Server和Graphics Server三个cartridge工具。

注意　安装的细节查询Oracle Developer Server的在线帮助。也应该查阅软件特定版本的发布文档中的修正信息。Cartridge配置进程是Oracle Application Server中标准管理的一部分，细节请查询 Oracle Web Server文档或Dynamic Information System，LCC的《Oracle Web Application Server Handbook》(Oracle, 1998)。也要注意Oracle公司说的多数客户使用静态HTML 方法，不是cartridge 方法。

动态HTML方法的好处包括HTML页的参数化和加载平衡。参数化的意思是用户不需要为每个应用编写不同的代码。页的动态特性使 cartridge代替模块名。如果想用表单服务器加载平衡，必须使用动态页和 cartridge方法。Oracle不支持静态HTML加载平衡。页的动态特性使服务器在运行时代替least-loaded服务器名。

B.3.1　用cartridge配置表单服务器

下面指导如何配置动态Forms Server的cartridge。

1) 从任务栏运行Oracle Application Server Manager，Start|Programs| Oracle Application Server | Oracle Application Server Manager。

2) 输入管理员用户名和口令(必须在安装Oracle Application Server时指定它们，在此处要使用与它们一致的用户名和口令)。

Oracle Application Server Manager是一个显示组件树和包含当前选择组件设置的独立图文框的Web页。

3) 假设用户位置的默认名为“ website40 ”。单击website40 Site打开树，找到website40

China-pub.com 下载

Site|Application。单击Add工具增加应用。

4) 在Add Application页，把Application Type域置为“ CWEB ”。

5) 单击Apply。在Add页，指定应用的名字、显示名和版本号。

6) 单击Apply，等待Success窗口出现。

7) 单击这个应用的 Add Cartridge按钮，显示Add CWEB Cartridge页。输入“ Forms60 Cartridge ”到Cartridge Name和Display Name域。Cartridge Name是cartridge 虚拟目录的名字，这个目录名作为运行应用的URL的一部分使用。在Cartridge Shared Object域输入Web cartridge DLL路径和名字(G:\ORANT\BIN\IFWKBC60.DLL，例如Oracle_Home为G:\PRANT)。置Entry Point(Share Object)域为“ form_entry ”。

8) 单击Apply，然后按下SHIFT键单击Reload工具，重新显示树。找到 web40site|Application|<app>|Configuration 文件夹(<app>是应用的名字)，然后单击 Web Parameters。在Listener List域指定一个接收者。

9) 单击Apply。找到web40site|Application|<app>|Cartridge Configuration文件夹，然后单击Cartridge Parameters。

10) 设置如下参数：

- code：这个输入项指向cartridge，“ oarcle.forms.engine.main ”。
- codebase：配置Web Server时指定的Java代码的虚拟目录。
- baseHTML：cartridge在运行时为建立动态HTML页用作模板访问的基准HTML文件的物理目录和文件名。这不是虚拟目录，而是真实的操作系统路径和文件名，如 G:\ORANT\webhtml\forms60cart.htm。虚拟html目录也应该指向这个目录。
- HTMLdelimiter：一个单一字符，在HTLM文件模板中用作替代指示，例如“ % ”。
- Minimum # of Instances：可能同时连接到cartridge的最小用户数，最好设置为0。
- Maximum # of Instances：可能同时连接到cartridge的最大用户数，例如100。
- achieve：在应用开始时下载的任何 JAR文件的路径和文件，包括 Oracle支持文件jfsrv60.jar。
- serverApp：指定字体和图标路径设置的应用类的名字，详见 14.2.1节“ 使用应用程序类 ”部分。

11) 单击Apply。现在，在html虚拟目录创建基准 HTML文件。从安装的版本复制文件($ORACLE_HOME\Tools\Devdem60\Web\CARTRIDGE.HTM)，改名为在baseHTML参数中指定的名字，并做如下修改：

- serverPort：设置Forms Server Listener端口(默认值为9000)。
- serverArgs：添加表单参数和用户参数。

可以在HTML模板中为cartridge设置任何参数，但是服务器从参数中取值，而不是从HTML模板中取值。下面是修改模板的一个例子 (回车符是为了使例子更清晰，不要在模板中输入它们)：

```
<HTML>
<HEAD><TITLE>Talbot Farms Web Applications</TITLE></HEAD>

<BODY><CENTER>
      <BR>Welcome to Talbot Farms.
      <BR>Loading application…
```

China-pub.com
下载


```
<APPLET CODEBASE="%LEASTLOADEDHOST%/codebase/"
      CODE="oracle.forms.uiClient. engine.Main"
      ARCHIVE="%LEASTLOADEDHOST%/jars/f60all.jar"
      HEIGHT=20 WIDTH=20>
<PARAM NAME="serverPort" VALUE="9000">
<PARAM NAME="serverArgs" VALUE="Module=%Module%">
PARAM NAME="serverApp" VALUE="default"
</APPLET>

</BODY>
</HTML>
```

这个例子把模块的名字参数化为% Module%，使用户添加一个模块名来运行。如果使用Forms Server加载平衡特性，则Forms Server用least-loaded服务器名代替%LEASTLOADED-HOST%参数。

B.3.2 用cartridge配置报表服务器

下面的部分指导如何配置动态Reports Server的 cartridge。

1) 从任务栏运行 OAS Manager，Start|Programs|Oracle Application Server|Oracle Application Server Manager。

2) 输入管理员用户名和口令(必须使用在安装OAS时指定的用户名和口令)。

OAS Manager是一个显示组件树和包含当前选择组件设置的独立图文框的 Web页。

3) 假设用户位置的默认名为“ website40 ”。单击website40 Site打开树，找到website40 Site|Application。单击Add工具增加应用。

4) 在Add Application页，把Application Type域置为“ D2KWEB ”。

5) 单击Apply。在Add页，指定应用的名字、显示名和版本号。

6) 单击Apply，等待Success窗口出现。

7) 单击这个应用的Add Cartridge按钮，显示Add D2KWEB Cartridge页。输入“ Reports60 Cartridge ”到Cartridge Name和Display Name域。在Cartridge Shared Object域输入Web cartridge DLL路径和名字(G:\ORANT\BIN\RWOWS60.DLL，例如Oracle_Home为G:\ORANT)。置Entry Point(Shared Object)域为“ rwows_entry ”。

8) 单击Apply，然后按下SHIFT键单击Reload工具，重新显示树。找到website40|Application|<app>|Configuration 文件夹(<app>是应用的名字)，然后单击Web Parameters。在Listener List域指定一个接收者。

9) 单击Apply。找到website40|Applications|<app>|Cartridge Configuration文件夹，然后单击Cartridge Parameters。

10) 设置如下参数：

- REPORTS60_WEBLOC：Web服务器查看报表输出文件的虚拟目录，例如/web_reports/。
- REPORTS60_WEBLOC_TRANSLATED：如果Reports Server和Web Server不共享一个文件系统时，cartridge传送Reports Runtime输出文件的物理目录，例如“ G:\ORANT\Reports ”。
- Minimum # of Instances：可能同时连接到cartridge的最小用户数，最好设置为0。
- Maximum # of Instances：可能同时连接到cartridge的最大用户数，例如100。

China-pub.com 下载

B.3.3 用cartridge配置图形服务器

下面的部分指导如何配置动态的 Graphics Server的 cartridge。与Forms Server和Reports Server不同，Graphics没有其他选项，只能与 Oracle Application Server一起运行Graphics Server。

1) 从任务栏运行OAS Manager，Start|Programs|Oracle Application Server|Oracle Application Server Manager。

2) 输入管理员用户名和口令(必须使用在安装OAS时指定的用户名和口令)。

OAS Manager是一个显示组件树和包含当前选择组件设置的独立图文框的 Web页。

3) 假设用户位置的默认名为“ website40 ”。单击website40 Site打开树，找到website40 Site|Application。单击Add工具增加应用。

4) 在Add Application页，把Application Type域置为“ D2KWEB ”。

5) 单击Apply。在Add页，指定应用的名字、显示名和版本号。

6) 单击Apply，等待Success窗口出现。

7) 单击这个应用的 Add Cartridge按钮，显示 Add D2KWEB Cartridge页。输入“ Graphics60 Cartridge ”到Cartridge Name和Display Name域。在Cartridge Shared Object域输入Web cartridge DLL路径和名字(G:\ORANT\BIN\GO60.DLL，例如Oracle_Home为G:\ORANT)。置Entry Point(Shared Object)域为“ GWWRBMAIN ”。

8) 单击Apply，然后按下SHIFT键单击Reload工具，重新显示树。找到website40|Application|<app>|Configuration文件夹(<app>是应用的名字)，然后单击Web Parameters。在Listener List域指定一个接收者。

9) 单击Apply。找到website40|Application|<app>|Cartridge Configuration文件夹，然后单击Cartridge Parameters。

10) 设置如下参数：

■ GW_TIMEOUT：显示在浏览器中保持打开状态的分钟数。

■ Minimum # of Instances：可能同时连接到cartridge的最小用户数，最好设置为0。

■ Maximum # of Instances：可能同时连接到cartridge的最大用户数，例如100。

China-pub.com
下载

术　　语

accelerator key　加速键

与菜单访问键一起连接到菜单项的键，提供对复杂的菜单访问按键的快捷方式。

action state　行为状态

内部行为的状态，至少是一次发出的转换；相当于在活动图表中的 Developer的进程。

Activex

基于COM的微软技术，可以实现在 Microsoft Windows环境下的应用程序和浏览程序中开发和重用组件控制。

activity diagram　活动图

描述处理的动态流程的UML图，除了基于状态过渡逻辑外与流程图相似。

alert　警告

显示信息或提出引起用户回答 yes-no类型反应的简单问题 (最多三个按钮，例如 yes，no，cancel)的窗口。

anonymous block　匿名块

一个不是子程序的PL/SQL块，因此没有名字。

applet　小应用程序

web浏览器下载并执行的一段代码，一个独立的软件组件。

application class　应用类

字体映射集和由特定字体和在 web浏览器中显示一个表单应用所使用的图标。

application window　应用窗口

在MDI用户接口标准中拥有多文档窗口的主窗口。

association　关联

两个或更多类对象之间的关系。

attached library　附属库

连接到另外同样个模块的pl/sql库，库的目的是使用户用 variations替换程序单元。

authentication　验证

确认用户身份的处理。

bar chart　条形图

在显示模块中以水平柱形图显示数据组。

base-table data block　基于表的数据块

映射数据库中表的数据块，数据块中的每格记录相当于表中的一行。

block　块

参见data block(数据块)和PL/SQL block(PL/SQL块)。

body　体

报表的中间部分，在重复帧和其他报表结构中包含报表数据的部分。

China-pub.com 下载

boilerplate text 样板文本

除了属于数据块的项之外，出现在画布上的字符串。

bookmark 书签

一种特殊的标签，显示在“特技”书签窗格中，就象 PDF文档中的超级链接点一样。

break column 分隔列

在报表中为列中每个不同值创建分隔的分隔组的列；缺省按列排序，使报表中相似的值在一起。

break group 分隔组

报表中处于记录组和其查询之间的组，使报表成为按级别间断点的分层结构。

built-in package 内部程序包

Oracle提供的作为Developer安装一部分的PL/SQL包程序。

built-in trigger 内置触发器

Developer结构提供的触发器。

canvas 画布

用来放置样板文本和项的窗口背景。

cartridge

用做软件系统接口的Oracle Application Server模块，使用户能够通过其Web浏览程序连接到Application Server；Oracle网络计算体系结构(NCA)的一部分。

CASE

计算机辅助软件引擎(Computer Aided Software Engineering)，在建立软件系统中辅助软件开发的工具。

category 分类

在命令访问控制方案中在分类里面再进行划分的局部分类。

chart 图表

(1) 数据的图形化描述；(2)在显示模块中显示基于数据模型的数据组或值的图形单元。

check menu item 检查菜单项

有切换选项的菜单项，如果切换打开则显示一个检查标记，如果切换关闭则没有检查标记。

class 类

在UML图中，一个矩形代表一个类型；对象是类的实例，类对应对象类型或表。

classification 分级

在命令访问方案中，识别标签信息的灵敏度的层次级别。

client/server 客户机/服务器

采用两台逻辑机器的配置软件的方式，一个客户机，一个服务器；对 Oracle Developer来说，服务器是数据库服务器，客户机是运行 Oracle Developer应用程序的客户端工作站。

column chart 列形图表

在显示模块中以垂直柱形图显示数据组。

column 列

描述表的每一行属性的数据库对象；相当于数据库对象的报表对象，与报表中的公式、

China-pub.com
下载

合计或占位列不同。

COM

公用对象模型(Common Object Model)，微软的一项OLE技术；允许通过标准接口对对象和数据访问。

combo box 组合框

一个结合弹出列表和文本项的控制件，使用户既能下拉一列文本项又能输入值。

commit 提交

事务处理成功结束命令。

concurrency 并发性

数据库管理员或应用程序允许在同一时刻多用户访问数据或运行应用程序。

console 控制台

表单中显示信息、记录数、lamps和任何其他状态报告的区域。

content canvas 内容画布

显示窗口基本内容的画布。

control block 控制块

不映射数据库表的数据块；通常控制块只有一个记录，记录中的项是可变的，在应用程序的子程序中可以引用(effectively ,package variables with data block scope)。

copy 复制

从另外一个对象建立一个Developer对象，而不在它们之间建立子集关系。

CORBA

公共对象请求代理标准(Common Object Request Broker standard)，在网络上提供对象模型和一组接口和建立分布式对象系统的工具的对象管理组 (Object Management Group)标准。

coupling 耦合

模块使用外部模块数据和运算的程度。

crosstab report 交叉报表

两列显示每列每个值的合计或组合的其他值的交叉列表；也称做距阵报表。

current record 当前记录

(1)数据块中最终用户定位游标的记录；数据块的所有行为都对当前记录起作用； (2)获得焦点的记录(参见focus焦点)。

cursor 游标

(1)指向焦点对象(参见focus焦点)的用户界面的抽象； (2)使程序员迭代处理查询返回行的SQL对象；(3)包含SQL语句(任何SQL语句)及其结果信息的Oracle对象。

data block 数据块

包含一组记录和一组包装每个记录的象的 Developer对象；参见基于表的数据块和控制块。

data file report 数据文件报表

用逗点或其他分隔符分隔的数据文件；使用数据文件报表来传递数据到其他工具。

data link 数据链接

在主-从报表中连接主组和从组查询的报表数据模型对象。

data model 数据模型

China-pub.com 下载

描述报表或显示数据结构的 Developer报表或图形对象，包括数据库数据镜像、数据对象键关系和组结构。

data type 数据类型

数据的分类，例如字符串、数字或日期。

database application 数据库应用程序

使用数据库管理系统(如Oracle 8)数据的程序。

database object 数据库对象

描述目标数据库中对象的Developer对象，例如表、索引、用户或其他模式对象。

database portability 数据库可移植性

在数据库管理下的应用程序的独立程度。

debug action 调试行为

报表或图形的PL/SQL代码中连接代码的断点。

decision 判断

在描述基于判断的控制流程的分层结构的活动图中的一个菱形。

declarative tool 说明工具

建立应用程序的工具，告诉系统想要什么而不是如何提供给用户。

default 缺省值

在建立对象时，运行系统赋给属性、变量或列对象的值。

default behavior 缺省行为

过程的默认行为，不要过程添加另外的代码，这样会改变其标准行为。

deploy 部署

分发一个产品到其用户。

dialog box 对话框

(1)显示选项或其他控制应用程序操作方式的常规模态窗口； (2)在MDI用户接口中独立于应用程序窗口的窗口。

discretionary access control 自主性访问控制

通过给使用应用程序及其数据的主体授权来控制对应用程序及其数据访问的能力。

display module 显示模块

包含一个或多个图表的Developer模块。

document 文档

显示应用“文档”对象的非模态窗口，通常是与应用程序主要部分相关的内容画布。

document window 文档窗口

在MDI用户接口中完全包含在应用程序窗口中的窗口。

domination 支配

在强制访问控制系统中，如果一个标签的分类等级大于等于另一个标签并且其分类是其他分类的superset，则一个标签支配另一个标签。

double-y chart 双y值图表

在显示模块中，为每个 x轴坐标值显示两个 y轴数据值；例如，可以用来显示两个不同的变量。

China-pub.com
下载

editor 编辑器

以文本编辑器窗口形式显示，用来在文本项中输入文本的 Developer对象。

entry transition 登录转换

在活动图中描述图中活动开始的转换。

event driven process 事件驱动进程

事件发生时执行的过程；事件可能是顺序的，或者事件作为用户任意顺序外部行为的结果发生，与标准顺序过程程序设计的风格非常不同。

exception 异常

PL/SQL运行系统在确定或遇到异常条件时产生的信号。

exit transition 退出转换

在活动图中描述图的结束行为的过渡；通过标注“ success ”或“ failure ”来显示行为的结果。

external query 外部查询

可以在应用程序间共享的独立文件中的 SQL文本；报表和显示可以包含以重用 SQL的Developer模块。

fire 引发

事件发生时执行一个过程；称为事件引发过程。

flexibility 适应性

不同需要的用户改编应用程序来达到其使用目的的程度。

focus 焦点

用户行为在给定时刻申请的单一 Forms对象；图形用户界面在任意给定时刻发生事件的窗口。

foreign key 外部关键字

从另外一个表引用一个表的主关键字的列，建立表间的连接。

form 表单

在一个或多个窗口中由一系列项组成以联机格式呈现数据的应用程序。

form letter report 表单字母报表

由空格填充的记录的样式文本环境数据，例如接受者的名字和地址。

form report 表单报表

每次显示一个记录的报表布局，通常是在单一页，但不总是这样。

formula column 公式列

在报表组中使用PL/SQL块计算的特定列，不是直接从数据库中取得数据的列。

frame 帧

在画布中包含布局属性，把一组块项作为一个整体的 Developer Forms对象；包含一组域和其他帧的布局的Developer Reports对象。

framework 框架

一起起作用来定义特定应用程序域需要的基本抽象的对象可重用系统。

FTP 文件传送协议

文件传送协议(File Transfer Protocol),从一个位置传送文件到另一个位置的 Internet协议。

China-pub.com 下载

function 函数

执行并返回一个值的PL/SQL子程序。

gantt chart 甘特图表

在显示模块中，显示描述开始和结束值的水平柱形图；通常描述一个一览表，在一览表中x轴代表时间，y轴代表任务组。

graph report 图形报表

包含图表或图形的报表。

graphic 图形

数据的图形描述。

group 组

在报表中把查询返回的记录作为记录重复组的报表模块结构；在报表中用分层结构描述记录的嵌套。

group above report 报表上组

由一组或多组列组成的表格报表，组帧显示在分组数据记录的上方。

group left report 报表左组

由一组或多组列组成的表格报表，组帧显示在分组数据记录的左方。

guard condition 保护条件

在活动图中由判断产生的连接到过渡的逻辑表达式；如果表达式结果为真，则控制流程通过过渡。

header 标题

报表的第一个部分，不包含报表数据，但包含样式文本、非重复的图形和其他类似的显示项。

high-low chart high-low图表

在显示模块中，用三个值(high、low、close)显示数据元素(例如证券交易的价格)。

horizontal toolbar 水平工具栏

在窗口的顶部显示工具按钮的画布。

HTML 超文本标记语言

超文本标记语言 (Hypertext Markup Language),广义标记语言 (Generalized Markup Language)的衍生语言，通过给文本和图形做标记来显示和运行 applet。

Interface 接口

访问一个系统(如数据库管理系统)的特定的方法。

Internet 因特网

提供计算机间的全球信息网(World Wide Web)和计算机间的国际因特网工作访问的全球广域网。

intranet 内部网

使用Internet技术，但通过防火墙或其他访问控制策略保护整个 Internet对网络访问的内部网络。

item 项

数据块中的一个单一值定义；每个项都有数据类型 (如字符串或数值)和根据项类型不同而

China-pub.com
下载

不同的一组属性。

Java

(1)Sun Microsystems的一种程序设计语言，大多数 Internet web浏览器能够在HTML基础上实现复杂的应用；(2)面向web 浏览器applets和组件的程序设计语言。

JavaBean

遵循Sun Microsystems的JavaBean规范的java模块。

key 键

键盘按键或其逻辑值例如Enter或F1。

key triggers 键触发器

当按下一个标准键(如F2)时引发的触发器。

label 标签页

在命令访问方案中的一个分级、分类和标志的组合，在每一个信息片上有一个标签，使得各片之间能够安全和清晰。

label ,list 列表标号

用于显示的列表值的标签。

lamp 指示灯

表单控制台的一个小的状态控制，只是某些特性的存在，如文本项的 LOV。

layout ,display 显示布局

显示模块中的图形元素。

layout ,report 报表布局

报表中的图形结构。

library 库

参见对象库或PL/SQL库。

line chart 折线图表

在显示模块中以数据的x、y值代表的点显示数据。

list 列表

描述一组值和值标注(参见列表值、列表标注)的Developer块项。

list of value 值列表

参见值列表(LOV)。

LOV 值列表

在对话框中显示一组值，并可以从中选择一个单一值的 Developer对象；可以连接LOV到一个记录组和一个项来检查输入到项中数据是否有效； LOV具有搜索功能，使得在大的 LOV中更容易找到目标值。

magic menu item 魔术菜单项

有内置行为的特定菜单项，与提供行为代码的菜单项不同。

mailing label report 邮件标号报表

某个区域中每页上的一系列有规律的重复记录。

mandatory access control 委托访问控制

通过给每个对象和会话关联一个安全标注来实现在数据库中为访问对象提供更强大的控

China-pub.com 下载

制的安全系统。

many-to-many association 多对多关联

一类的多个对象与另一类的多个对象的关联。

margin 边界

报表中文本边界以外的报表部分。

marking 标志

在命令访问方案中一种在分类和分级中更多的组件片，给出在不同情况下的数据之间的关联。

master-detail form 主从表单

把表单分割为父记录和几个子记录。

Master-detail report 主-从报表

一个主记录和一个或多个子记录一起显示。

master-master report 主-主报表

两组不相关的记录一起显示。

matrix report 矩阵报表

两列显示每列每个值的合计或组合的其他值的交叉列表；也称做交叉报表。

maximize 最大化

把窗口扩大到整个屏幕或在多文档界面中把窗口扩大到应用程序窗口大小。

MDI

多文档界面(Multiple Decument Interface)。

menu 菜单

一列文本项，可以通过单击文本首行 (菜单名)来访问。

menu access key 菜单访问键

与选择一个菜单项有同样效果的按键，通常包括一个转义键如每个子菜单用 ALT加上一个字母。

menu item 菜单项

显示另外一个菜单(子菜单)的菜单项。

menu module 菜单模块

在应用程序中描述一个或多个菜单并包含菜单和菜单项定义及菜单执行代码的 Developer 模块。

minimize 最小化

使窗口缩小成为图标。

mixed chart 混合图表

在显示模块中数据同时以点和线显示。

moat 护城河

在图形用户界面上的按钮周围的边界线，它表示窗口中的按钮是缺省按钮，当按下回车键时执行按钮的功能。

modal window 模式窗口

在应用程序中，在其他任何窗口做任何事之前需要用户响应和释放的窗口。

China-pub.com
下载

modeless window 非模式窗口

(1)在应用程序中允许用户在释放非模态窗口之前转移到另一个窗口的窗口； (2)用户能够在任何时候聚焦的窗口。

module 模块

Developer应用程序最高级的组件；运行时程序运行模块，模块集中了表单、报表、菜单、代码库、对象库等等所有的对象。

move 移动

在Object Navigator中改变Developer对象的位置，可以在模块中和模块间。

Multiple Document Interface 多文档界面

图形用户界面的一种窗口标准，把应用程序组织到一个主应用程序窗口，在主应用程序窗口中包含多个文档窗口。MDI窗口有应用程序菜单；也显示控制台。

multiplicity 多重性

参与关联的对象数目；参见一对一关联、一对多关联和多对多关联。

named visual attribute 命名的可视属性

可以从另一个对象引用的Developer对象的一组显示属性；参见属性类。

navigability 导航性

在图形界面中把焦点从一个控制转移到另一个控制的能力。

navigation 导航

把焦点从一个对象移到另一个对象 (参见焦点和导航性)。

non-updateable view 不可更新视图

视图的选择列表中有连接或表达式，使 Oracle不能通过视图更新其基础表。

null menu item 空菜单项

没有关联代码的菜单项；分隔符是一个空菜单项，也可以通过设置命令方式为空使任何菜单项为空菜单项。

object group 对象组

使用户能够打包可重用对象，以备以后复制或引用的 Developer对象；在一个标题下集中了一组对象。

object library 对象库

Developer表单对象程序包，以备在其他应用程序中重用的 Developer模块。

object type 对象类型

定义一类对象的结构和行为的Oracle 8对象；可以使用对象类型建立表或表中的列。

ODBC

开放数据互联(Open Data Base Connectivity)标准，为任意数据库管理系统编写一个驱动程序，提供统一的SQL语言和程序设计接口的微软事实上的标准。

OLE 对象链接和嵌入

对象链接和嵌入(Object Linking Embedding),把文档嵌入到其他文档中的微软技术。

one-to-many association 一对多关联

某一类的一个单一对象与另一类的几个对象的关联。

one-to-one association 一对一关联

China-pub.com 下载

某一类的一个单一对象与另一类的一个单一对象的关联。

open system 开放系统

允许使用其他厂商提供的工具的系统。

Operating system (OS)portability 操作系统可移植性

工具和数据库管理本身的可移植性。

package 程序包

(1)包含其他PL/SQL对象的PL/SQL程序单元，包括子程序、类型和属性；(2)描述一种分类(如类和其他包)的一个UML对象；在作用持久的数据库图中，程序包描述一个模式。

parameter 参数

模块定义的表单、报表或显示数据对象，可以通过模块的运行系统赋值；参数可以在运行时定制一个模块，提高不同条件下模块的可重用性；参见系统参数和用户定义参数。

parameter form 参数表单

为输入项显示的表单，可以在其中为一组参数赋值。

parent class 父类

一个子类引用的对象。

PDF

可移植文档格式(Portable Document Format)，一个Adobe System标准文档格式，用语在用户之间传输印刷的文档。

pie chart 饼图

在显示模块中，把数据组显示为环状饼中的一片。

PL/SQL

一种块结构的程序设计语言，包括 Oracle SQL，顺序、分支、循环控制语句和其他现代的结构化程序设计概念，例如子程序、包和异常处理等。

PL/SQL block PL/SQL块

一系列PL/SQL程序设计语句，包含一组数据声明或异常处理；一个块有可能是匿名的，或者是子程序(函数或过程)。

PL/SQL library PL/SQL库

包含一组PL/SQL程序单元的 Developer对象，把库连接到要连接的地方就可以重用。

PL/SQL menu item PL/SQL菜单项

有执行行为的PL/SQL代码的菜单项。

placeholder column 占位列

一种报表列，占位列有触发器、公式或用户退出填充，而不是取数据库值或标准合计。

poplist 弹出列表

一个下拉文本项列表，可以通过紧挨者列表的方框中的一个小箭头激活，引起列表在项的附近弹出。

pop-up menu 弹出菜单

一个浮动菜单，当用鼠标右键在画布或项上单击时弹出。

portability 可移植性

在不同系统上运行的能力；参见数据库可移植性和操作系统可移植性。

China-pub.com
下载

post 记入

通过一系列INSERT、UPDATE和DELETE语句把表单中的改变写入到数据库，而不提交事务。

primary key 主关键字

表中的一列或多列的值对于行是唯一的。

privilege 权限

给特定的用户授权对数据库对象的行为。

procedure 过程

执行而没有返回值的PL/SQL子程序。

process 进程

任务或子进程，事件，触发器的顺序。

program unit 程序单元

PL/SQL包或子程序。

propagation 传播

在PL/SQL程序中产生异常时，把控制移交给异常处理器的过程。

property 属性

类所有对象相关的特性；对应于对象类型的一行或表。

property class 属性类

由任何种类属性组成的 Developer对象；参见命名的可视属性。

query 查询

返回数据值的SQL语句，是报表和显示的基础；参见外部查询。

query by example 例子查询

用户通过提供描述期望数据的例子记录的值 (和逻辑条件)建立的查询。

radio button 单选按钮

一组小的圆形按钮每个按钮有一个标签；选择一个按钮的同时关闭其他按钮，单选按钮建立了一个互相独立的选择。

radio menu item 单选菜单项

描述一个选项的几个选择之一的菜单项，其他选择在菜单项的附近。

record group 记录组

(1)描述与表类似的特定数据结构的 Developer对象，有行和列；(2)描述查询返回数据的Developer报表数据模型对象。

record type 记录类型

一系列数据域，数据的一行。

recovery 恢复

数据库管理系统在系统崩溃时保护数据的能力。

referential integrity 参照完整性

数据库和/或应用程序确定什么时候一行引用另外一行 (第二行应该存在)的能力。

relation 关系

描述主-从关系的 Developer对象，用关系表达两个数据块间的影射。

China-pub.com 下载

repeating frame 重复帧

包含重复的域顺序的Developer报表对象，每个重复来自报表组行。

report 报表

一个数据的页导向显示。

report module 报表模块

定义报表的 Developer模块。

reusability 重用性

在不同环境下使用对象的能力。

role 角色

(1)类之间的关联的一个方面；(2)包含一组权限并被授权给某些 Oracle用户的Oracle安全特性。

rollback 回滚

事务处理的不成功结束命令。

rowid 行标识符

一个唯一的Oracle内部行标识符。

running total 运行合计

对整个报表合计的总数。

savepoint 保存点

Oracle的一个特性，在事务处理中标记一点，这样可以在必要的时候回滚而不用回滚整个事务。

scatter chart 散点图表

在显示模块中以x、y值所代表的点的形式显示数据值。

schema 模式

有特定的一组表、视图和权限的对象。

scope ,trigger 触发器作用域

引发触发器的对象的位置；作用域由拥有这个触发器的对象和任何属于这个对象的对象组成。

scroll bar 滚动条

使最终用户可以移动一组记录或在可视显示区域 (计算机屏幕、另一个窗口、应用程序窗口或其他)内移动窗口的用户界面。

security 安全性

数据库管理系统或应用程序保护数据或应用不被未授权使用的能力。

security clearance 安全许可

被授权读写信息的委托访问控制系统中标注的范围。

separator menu item 分隔符菜单项

分隔其他菜单项的菜单项，分隔符菜单项没有关联的行为。

sequence 序列

按要求产生的一个有顺序的整数的 Oracle对象。

Servlet 程序

China-pub.com
下载

Application Server运行的一段代码；与运行在客户端web浏览器的applet不同。

SmartTriggers 智能触发器

Form Builder的一个特性，可以确定大多数所选择对象可能要连接的触发器，通过一个弹出菜单可以选用它们。

specification 说明

过程或函数参数或函数返回值的类型。

stacked canvas 堆栈式画布

一个覆盖在其他画布上的画布，显示条件的或可分离的内容。

standard function key 标准功能键

一个程序定义的键盘按键或逻辑等价物，通常是从 F1到F12或F24。

static structure diagram 静态结构图

描述一个系统的静态结构的UML图，主要由类型和类型间关联组成。

status bar 状态栏

参见控制台。

subclass 子类

(1)引用另外一个对象(父类)并继承其属性和关联的对象；(2)建立这样一个对象。

submenu 子菜单

从另外一个菜单显示的菜单。

subprogram 子程序

过程(没有返回值的子程序)或函数(有返回值的子程序)PL/SQL程序单元。

summary column 汇总列

报表组中的特定列，Developer通过聚集报表数据计算合计列，例如总和 (totals)和记录数(counts)。

system parameter 系统参数

Developer自动定义的参数，例如报表目标(report destination)。

tab page 标签页

在标签画布上显示的个体标签布局。

tabbed canvas 标签画布

在一系列标签组成的重叠的画布上显示窗口基本内容的画布。

table chart 表格图表

在显示模块中以表格格式显示数据。

tabular report 列表式报表

数据的简单列表。

tag 标记

标记语言的一个元素，描述跟随的文本或图形的性质；HTML使用标记为在web浏览器中显示标出文本和图形，另外也设置选项和在浏览器中运行 applet。

template 模板

可以使用其基本选项和内容格式化一个模块的对象；表单、报表和图标都有不同类型的模板。

China-pub.com 下载

template ,chart 图表模板

可以用来建立几个同样格式图表的定制的选项，可能在同一个显示模块中。

three-tier 三层

应用程序的一种配置方式，应用程序主要运行在 Application Server，Application Server依次访问数据库服务器，数据库服务器通过客户端工作站只管理用户接口函数；三层分别是瘦客户机、Application Server和数据库服务器；与两层的客户机/服务器不同。

timer 定时器

一个显示模块元素，象一种控制显示的闹钟；允许特别效果、重复查询等等。

t-list t-列表

在滚动列表中显示所有文本项的列表框，象一个独立的画布，但在表单中包含一个单一的域。

toolbar 工具条

沿着窗口的上边或左边的一条图标。

trailer 报表尾

报表的最后一个部分，不包含报表数据，但包含样式文本、非重复的图形和其他类似的显示项。

transaction 事务处理

工作的逻辑单元；以数据相关行为开始，以提交或回滚结束。

transition 转换

在图中从一个行为状态到另一个行为状态的控制流程或判断。

trigger 触发器

(1)连接到Developer对象的匿名PL/SQL块，当这个对象的特定事件发生时执行它；反映触发PL/SQL块的事件类型的触发器名字；(2)在服务器上与Oracle数据库触发器对应的数据库对象；(3)表的特定事件(插入、更新或删除)发生时执行一个PL/SQL块的Oracle模式对象。

trigger ,report 报表触发器

报表处理中(报表前和报表后、每页之间、参数表单之前或参数表单之后)定义明确的点执行的PL/SQL代码块。

two-tier application 两层应用程序

运行在工作站并访问服务器的应用程序。

UML

统一建模语言(Unified Modeling Language)，通过图描述对象导向设计的对象管理组标准系统。

user-defined parameter 用户定义参数

可以建立和初始化的参数，与 Developer自动建立和初始化的参数不同。

user-defined trigger 用户定义触发器

自己命名而不是用标准触发器事件命名的触发器，可以在 PL/SQL代码中通过过程调用引发该触发器。

validation 确认

确认对象是否满足在设计表单时设置的所有约束条件。

China-pub.com
下载

value ,list 列表值

列表项的一个元素。

vertical toolbar 垂直工具栏

在窗口的左边显示工具按钮的画布。

view 视图

覆盖画布一部分或全部的窗口中的矩形。

web browser Web浏览器

一种轻型软件系统，它在客户机上运行，通过 URLs访问覆盖网络的应用程序，常使用HTML(超文本标记语言)和从服务器下载的java applet程序去执行应用程序。

window 窗口

由图形用户界面(GUI)平台支持的应用程序的矩形显示区域。

World Wide Web 全球信息网

通过因特网的超文本传输协议(HTTP)使超文本标记语言(HTML)页面互联的全世界系统。

China-pub.com
下载

第一部分　建立应用程序

第1章　介绍Developer

Oracle Developer是一个综合的高度有效的开发环境。它可以为因特网或客户机 /服务器系统建立可扩充的数据库应用程序。 Oracle Developer Server是一个Application Server，专门用来配置和优化因特网的Oracle Developer应用程序。本书为你讲授在开发功能强大的、可重用的、有效的、用于访问数据库中的数据应用程序时所需要了解的一切。

1.1　综述应用程序的建立

应用程序是执行某个任务的计算机程序。数据库应用程序是在数据库管理系统 (如Oracle 8)中管理数据的程序。应用程序显示数据，通过添加和修改将数据移入和移出数据库。

建立应用程序系统有许多方法。可以使用各种程序语言，如 COBOL、C、C++或Java。程序语言很多，每个程序语言都有其优缺点。用这些语言编程序提供了很大的灵活性、控制权和性能。但是必须努力工作才能充分利用这些特点。甚至要更加努力工作才能确保应用程序在日益广泛的应用中使用。这种编程方法容易发生错误。不精心编制的程序容易失败，降低而不是加速程序执行的速度。

Oracle Developer是一种说明性工具。用它建立应用程序，主要是告诉数据库要做什么，而不是告诉它如何做。只要用几条语句或插入几个参数，就可以生成可扩充的、能够在 Web上运行业务的应用程序。在PL/SQL中另外编制一些代码，就可以做许多想要做的事情。

当前，对因特网开发的应用程序包含有带菜单、工具栏、对话框和窗口的丰富的图形用户界面，不管应用程序的对象是什么，都可以采用窗口显示。我们感兴趣的三种对象是表单、报表和图形。 Oracle Developer分别为这三种对象提供了模块： Oracle Form(表单)、Oracle Report(报表)和Oracle Graphic(图形)。

如果不需要高效的配置应用程序，则开发一个应用程序不是很困难。以前，使用一个应用程序就意味着要找出一种方法使用“ 人工传递网络 ” (SNEAKERNET)将应用程序安装在商店中所有的用户机器上。现在，采用内部网、外部网和因特网，使用起来非常容易；但扩大则比较困难。只要某个地方有运行应用程序的服务器，则任何人只要有 Web浏览器，就可以运行现在的应用程序。 Application Server(如Developer Server)的任务是执行服务器的应用程序的业务逻辑。这个服务器连接到浏览器上显示表单、报表和图形。

1.1.1　表单应用程序

要了解Oracle Developer，需要了解一些与应用程序对象有关的术语。

表单应用程序(form application)是一种以联机方式的格式提供数据的应用程序。它由一系列或多个窗口上部署的项目组成，这些项目可能是可输入的字段、图形、按钮、复选框或任

China-pub.com 下载

何数目的对象。这里给出一个表单应用程序的示例：

如图1-1所示，表单提供了观看数据库信息的一个很好的方式。可以将数据输入到表单的域中，也可以修改域中的数据，这取决于所使用的应用程序。

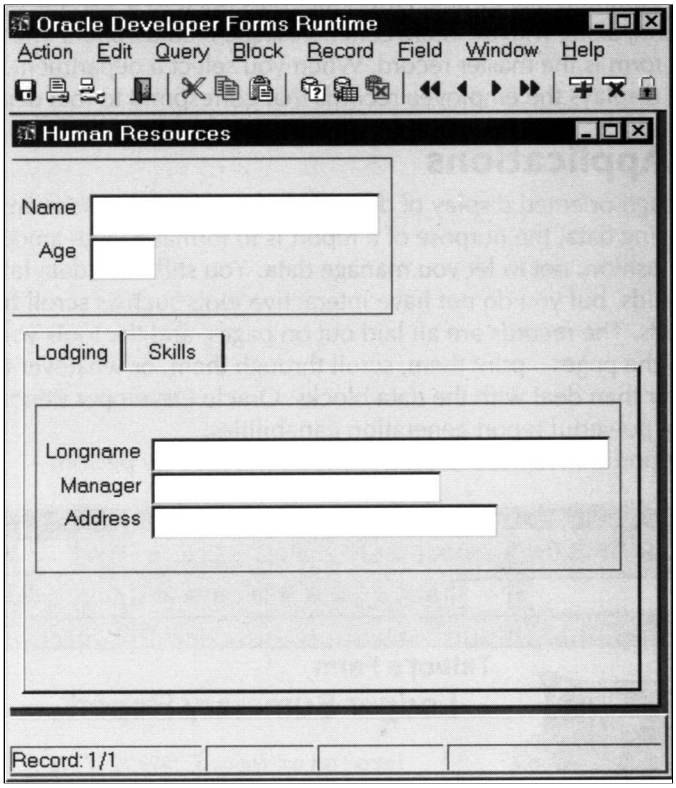


图1-1 表单应用程序示例

依照应用程序所要完成的任务和运行它所要求的体系结构而定，一个应用程序可以只有一个表单或也可以有多个表单。如文件的格式一样，表单的设计也是一种艺术：必须以正确的格式显示信息，否则会难以使用并且容易出错。将一个非常大而且复杂的表单分成几个部分，是减少其复杂性的常用方法。在联机应用程序中，也采用同样的处理方法。简单的表单对应于记录类型，它是一系列数据字段 —— 单行数据。表单可以一次显示单个记录，或者可以同时显示几行。表单中的工具可以管理记录和显示信息的交互操作 (查询、插入、更新、删除、滚动等)。

有一种特殊类型的表单叫做主 -从(master-detail)表单，它自动地管理父记录与几个子记录的关系。这种情况很普遍，所以 Oracle Developer将它和更加普遍的单个记录一起作为建立表单的专项目选择。表单的各个部门是主记录，当选择一个部门时，表单自动地显示与这个部门对应的雇员情况。

1.1.2 报表应用程序

报表面向页面显示数据。表单为数据管理提供了交互式的工具，而报表的目的是以一种可读的形式来格式化大量的数据，而不是管理数据。报表数据仍然在记录和字段中部署，但

China-pub.com
下载

是没有交互工具(例如滚动条)去处理记录。记录在页面上部署，处理页面的工具有页面打印、翻页和一些格式允许的操作，但不处理数据。 Oracle Developer Reports Builder有很强的报表生成能力。

图1-2是一个报表，表示按照人员分组的Ledger项目：

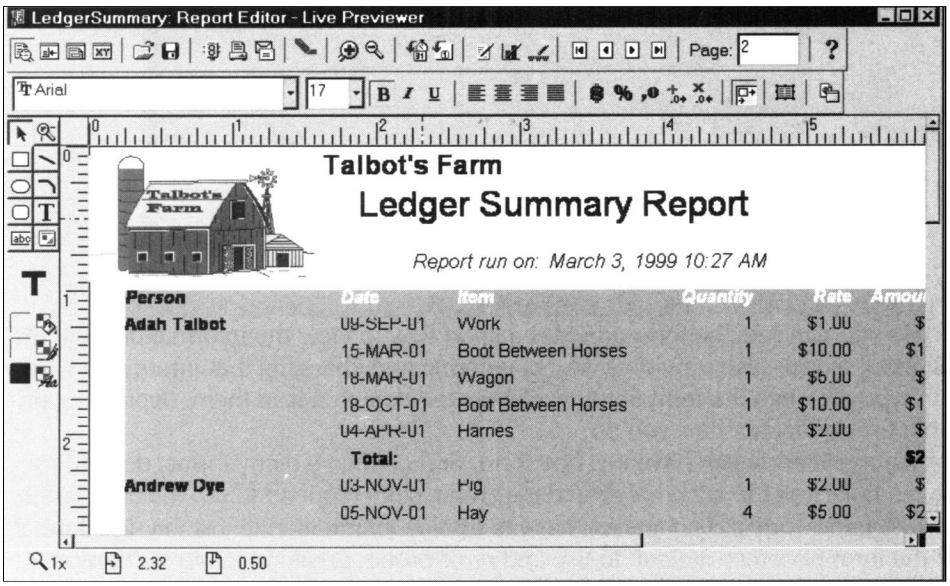


图1-2 Ledger Summary报表

这个报表和写在纸上的 Ledger十分相似。它有页面脚注、标题和所有面向页面报告的装饰。

Oracle Developer提供许多不同种类的报表，而且单个报表可以组合成不同的类型。

- tabular(表格)：简单表。
- mailing label(邮件标签)：在每页的某个区域内的一系列有规律的重复的格式化记录。
- form letter(格式信笺)：一种占据了整个文本空白处的样板文字，它被安排在数据记录的周围，比如接受者的名字和地址。
- master-master(主-主)：两组显示在一起的没有关系的记录。
- master-detail (主-从)：一个主记录带有两个或多个有关的从属记录，显示在一处。
- matrix or crosstab(矩阵或交叉)：一个两列的十字表格，表示某些集合或者其他来自各列值的组合值。
- datafile (数据文件)：使用一个逗号分隔的或其他的各种分隔的数据文件，使用这种文件将数据传输到其他工具(通常有比较容易的办法)。
- graph (图)：包含图表或数据图表除此之外或是取代显示数据本身。

注意 Oracle Reports完全能独立使用。事实上，Oracle将Oracle Reports当作独立的产品和Oracle Report Server一起出售。

1.1.3 图形应用程序

图形是数据的形象化表示。术语 chart 指的也是这种表示。在它们所有的变化中，折线图、

China-pub.com 下载

饼图和条形图都是使数据形象化，而不仅仅是显示数据。 Oracle Developer提供了一套高级工具，用来生成可以独立应用的图形应用程序和用来在表单及报表中包含图形。

图1-3是一个用饼图表示的图形应用程序。

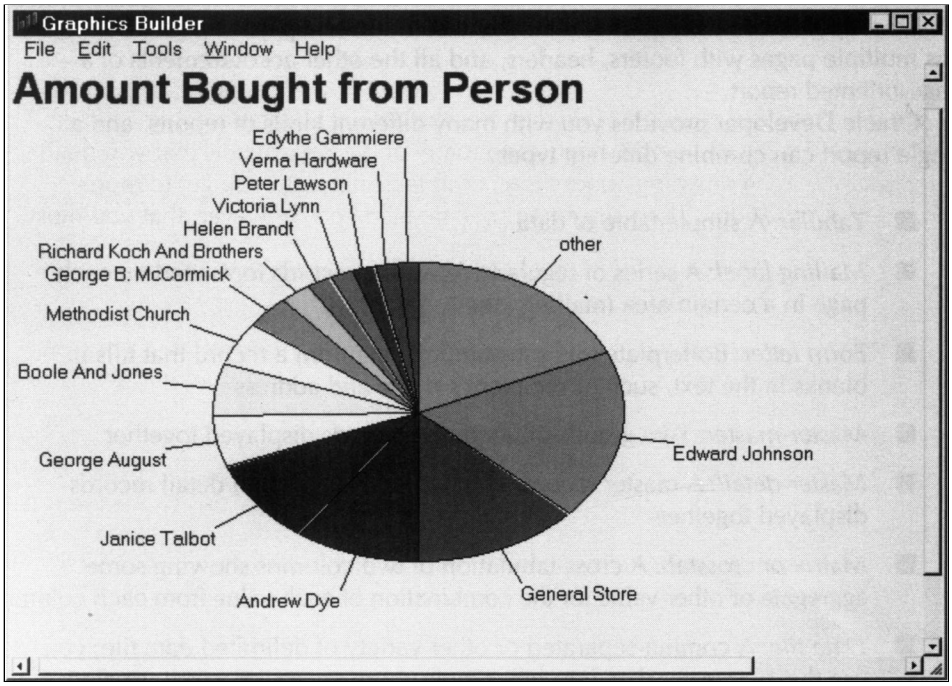


图1-3 饼图图形

甚至可以使图形被激活，这样用户就可以操纵它们。例如，用一个图形表示某种数据的总合计，比如推销员收入的合计。可以为用户产生这样一种能力，下钻 (drill down)查看按城市分解的每个推销员的细节图。当将图形嵌入表单应用程序时，这种能力特别强。有关下钻(drill-down)图形的详细内容，请参见第8章。

1.2 数据库应用程序有什么不同

是什么使得数据库程序如此特殊？它与通常的应用程序有什么不同？

数据库管理引擎对每况愈下的情况提供数据保护 (恢复)，支持多用户(并发)、多个应用程序和参照完整性，防止未授权使用数据(安全性)，而且将用户和在特殊平台上管理数据的细节隔离开(可移植性)。SQL和关系数据模型或对象关系数据模型提供了一个很强的、可靠的数据访问的范例。数据库管理引擎和所使用的数据特性和有效维护数据的其他工具紧密地连接和结合在一起。

Oracle Developer自动地处理数据管理的细节。事实上， Oracle Developer处理如此多的细节使得你可以不必学习与这些细节有关的任何东西，或者至少要在高级的应用程序中才需要学习。但是，为了有效地使用Oracle Developer，必须了解其中的一些特殊的组件。

如果想继续研究Oracle Internet Database Server(Oracle的因特网数据库服务器)在现场是如何工作的，请参看 Oracle 的文档资料或有关 Oracle数据库管理和编程的书籍。比如 George Koch和Kevin Loney著《Oracle 8完全参考手册》，或Kevin Loney著《Oracle 8i数据库管理员手

China-pub.com
下载

册》一书。

事务是工作的逻辑单元。事务从与数据有关的动作开始，用一个成功的终止命令 Commit (提交)或一个不成功的终止命令 rollback(回滚)结束。这些限制本质上建立了数据操作的顺序，如果需要做，可通过回滚数据库的变化来撤消操作。

Oracle Developer应用程序需要操心的是当事务结束时给数据库引擎发信号，通常用 SQL COMMIT WORK 语句或ROLLBACK WORK语句，这对应用程序设计有某些暗示。

必须做的第一个选择是，决定用户是使用提交还是回滚。用菜单、按钮或工具按钮进行选择，这是对最终用户在数据库工作进程上总的控制。他或她可以实现一个小的变化或者许多大的变化 —— 凡是适当的都可以。

接下来要做的选择是决定用什么构成一个逻辑单元。即使是允许最终用户控制提交或回滚，应用程序仍然需要在适当的时刻提供这些选择。例如，可以在删除一行和删除依赖于这一行的各行后允许提交或回滚。如果允许终端用户在删除以后和在相关的删除之前提交，那么就违反了参照完整性(在稍后讨论)。要考虑在磁盘崩溃事件中数据库引擎应该恢复什么。如果要恢复的内容包含了多个操作，那么应将所有有关的操作都组装到一个事务中。

Oracle Developer好的一面是它完全了解所有这些问题，而且自动地处理它们。它自动地在应用程序逻辑上的适当时刻处理终端用户的提交和回滚请求，它在界面上自动产生提交和回滚命令，它提供了一种方法将每个事务支持的操作作为其应用程序结构的一部分包含在内。

参照完整性是数据库确认什么时候一个数据行引用另外一个已存在的数据行的能力。Oracle和其他的数据库管理系统通过表上的完整性约束或通过启动公共事件，比如插入、更新和删除的触发器来管理数据库的大多数参照完整性。在可能的情况下，应该将这种确认移到数据库服务器，以便所有的应用程序都共享它。保留应用程序的有效代码。

每个表(table)有或者应该有一个主关键字。主关键字是表中带有唯一标识的列或多列。关系数据库的本质是通过到另一个表的主关键字的数据引用来表示表之间的逻辑关系。引用的列是外关键字，而且它指的是外表的主关键字。大多数表连接与外关键字和主关键字的匹配有关。

你将会看到维护这些数据指针的完整性是如何重要。在关系数据库中，指针是数据，而不是那种可以被搞坏的物理指针。还有一个问题是保证当出现数据引用时，确实映射到某地址的主关键值。对于某些关系还有一个相反的问题是要确认对每个主关键字，存在一个引用它的外关键字。最后必须确认主关键字能够唯一地标识表的行。

可移植性是指在不同的系统中运行的能力。它适用于操作系统，也适用于窗口系统、应用程序框架和任何其他服务平台，其中包括数据库管理平台。大多数应用程序必须以几种不同的方法处理可移植性。对 Oracle Developer应用程序的要求是能够在不同的数据库管理系统或在不同的操作系统运行同一个应用程序。

数据库可移植性是指如何使应用程序与它的基础的数据库引擎无关。通过界面或界面组这种专门的方法访问数据库管理程序可能是个问题。在这里，必须要决定的是，应用程序对于多个数据库平台需要的运行范围，用户要求应用程序在 Oracle 8、Informix和Sybase中运行吗？如果要求，应该立即果断地限制在标准的 SQL上，可以说这已经是最低的标准了。

在这种情况下，ODBC可以有所帮助。但是它的主要作用是使得可以从市场上得到的不同数据库的接口标准化。ODBC(Open Data Base Connectivity standard，开放数据库互联标准)提

China-pub.com 下载

供了统一的SQL语言和编程接口，供那些愿意对它写一个驱动程序的数据库使用。对于各个有影响的数据库，都有厂家和第三方的 ODBC驱动程序(有关在Oracle Developer中使用ODBC接口的细节请参看第15章)。通过ODBC，可以在运行时转换数据库，而不需要重新编译代码。

操作系统(OS)的可移植性是工具和数据库管理程序本身的可移植性 (原文如此，但这个说法是不对的。 —— 译者注)。如果要在不同的平台上运行应用程序(例如，在Internet、Windows、Macintosh或者UNIX上运行)，则需要将应用程序转向不同的数据库管理程序，或者是数据库管理程序和工具必须能够用同样的接口在不同的操作系统上运行。当 ODBC驱动器在不同的操作系统和各种数据库管理系统中得到利用时， ODBC在逐渐地发挥它的作用。

在使用Oracle Developer时并不一定需要了解SQL。当进入到产品的较高级的使用时，才会发现这些知识有用。如果还不懂得 SQL，那么应该进行训练，或者去读一本有关这个语言的书，比如Koch和Loney著《Oracle 8完全参考手册》，或者Groff和Weinberg著《SQL：The Complete Reference》(Osborne/McGraw-Hill,1999)。

最后，Oracle Developer允诺能够使高质量的可扩充的应用程序在三级环境中工作。两级的应用程序在工作站上运行并访问数据库服务器。三级应用程序主要运行在 Application Server上，由Application Server去访问数据库服务器，通过客户机工作站只管理用户界面的功能。使用Developer Server，可以为大量客户机及巨大的数据库建立健壮的、高性能的应用程序。

1.3 因特网数据库应用程序有什么不同

因特网在应用程序开发方面所取得的进展是令人惊奇的。只在短短几年的时间里，因特网就完全改变了我们关于应用程序的想法。和其他的数据库应用程序一样，因特网计算要求应用程序开发环境比以前所要求得更多。因特网数据库应用程序必须是：

- 可靠的。
- 安全的。
- 高度可用的。
- 可扩展的。

可靠是很重要的。因为许多公司现在依赖因特网应用程序来运行它们的业务。一个公司要使其产品或服务可以从因特网上得到，必须保证传递这些产品和服务的应用程序工作可靠。在这样的竞争环境中，发生错误会使公司丢失信誉和利益。

许多因特网应用程序处理钱财、私人信息或商业秘密。这种应用程序要求十分安全，以保护具有价值的数据和系统用户秘密。

电子商务的新领域对在因特网上使用的数据库有很严格的要求。这些应用程序必须 24小时可用。你可以睡觉，但是你的客户，不管在世界的什么地方，可能并不睡觉。所以数据库应用程序也不能睡觉。

最后要说的是，最近几年很少向我们显现出的是对扩展性的要求。没有人能够想象得到因特网的急剧增长。当公司将它们的产品和服务上网时，经常发现他们低估了对访问的要求。如果因特网数据库应用程序不能处理用户数量的增长，那么很快就会处于困境之中。

这些要求就像是公司内部使用的企业范围的应用程序要当作企业外部的世界范围的应用程序来使用一样重要。Oracle Developer为这些应用程序提供了很强的解决办法。这些应用程

China-pub.com
下载

序对开发工具的要求很高：网络可能对数据流形成瓶颈，服务器提供的性能可能很差，软件的层次和扩展性的要求可能超过了开发者的能力。

Oracle Developer Server是一个专门用于和优化Oracle Developer应用程序的应用程序服务器。它允许将新的和现行的应用程序应用到因特网上，而不需要修改应用程序的代码。

一个好消息是Oracle Developer Server可以自动在网络上进行调整和执行，并且自动递送本机服务和内部函数性以确保可靠性、安全性、可用性和可扩展性。

Oracle Developer提供了一个创建可扩展的、执行关键任务的数据库应用程序所必须的集成开发环境。Oracle Developer Server为把Oracle应用程序应用到因特网上服务提供了一个专门的优化的Application Server。本书将用几部分内容指导你如何使用 Oracle Developer和Oracle Developer Server，教程的重点将放在产品的基本特性上，还将对产品的细节提供综合参考。将个别指导的内容和参考资料结合起来，就是一个利用 Oracle Developer建立高性能数据库应用程序的完全手册。

开发过程讲述到此为止。第2章讲述在Oracle Developer应用程序中创建对象的细节。

China-pub.com
下载

第2章　Oracle Developer对象综述

Oracle Developer有哪些专门的工具？本章综述各种可以用来生成应用程序的工具和对象。这将为以后各章的学习打下基础。图 2-1展示了Oracle Developer系统的总体结构。

Project Builder是使用Oracle Developer模块工作的主要接口，一般不需要使用它。但是，如果要用两个以上的表单工作时，将模块组织到工程之中将会使你轻松一些。可以为每个工程的Oracle Database Server建立一个缺省的登录，以免出错或因为访问错误的开发数据库而浪费时间。可以用单击按钮来生成或重新生成模块，而不必令人厌烦地去逐个通过每个模块。可以设置工程工具条以访问所有的工具，而且可以书写宏，自动生成各种过程。

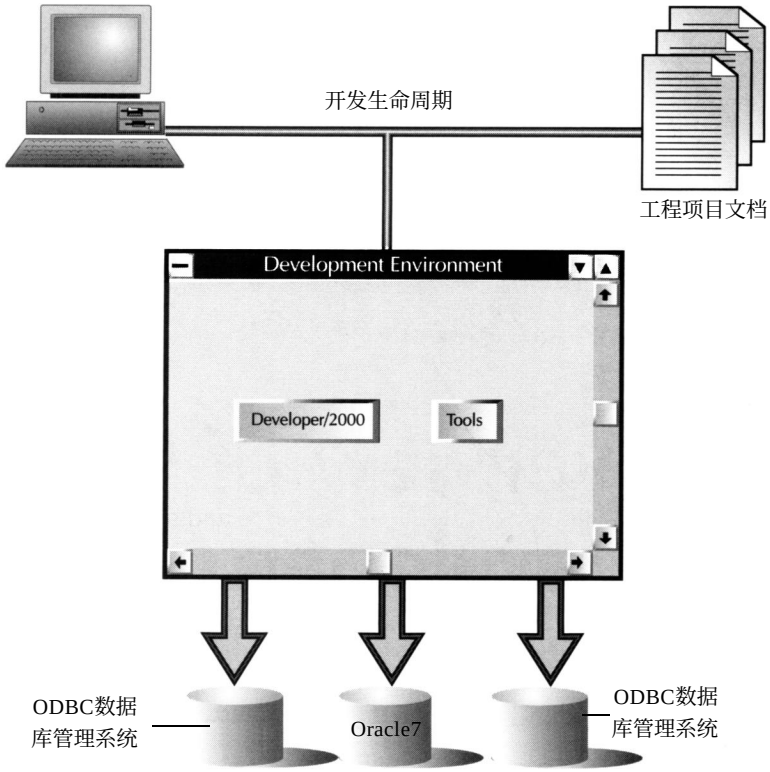


图2-1　Oracle Developer结构

Project Builder能够组织程序和自动生成程序。然而，创建 Oracle Developer应用程序的实际工作是在单个生成器 —— 表单、报表和图形生成器。这些工具可以创建各种应用程序模块：表单集、报表集、图形集。这三个工具中的每一个都有自己相应的编译器。这些编译器在编译模块时不用整个生成器环境的开销。还有一些有所帮助的工具： Procedure Builder、Query Builder、Schema Builder和Translation Builder。Procedure Builder帮助创建存储在Oracle数据库服务器上的程序，或者创建用户的 PL/SQL代码。Query Builder帮助创建使用图形界面的SQL SELECT语句。Schema Builder帮助创建关系数据库的模式。转换生成器定位表单、报表

China-pub.com
下载

和图形，并且转换所使用的各种字串。

这些组件构成了建立应用程序的 Oracle Developer。在准备部署应用程序时，安装 Oracle Developer Server，它由Forms Server、报表服务器和图形服务器组成。这三个服务器和 Oracle Application Server一起工作，使得可以将应用程序部署到因特网 (或本地内部网)上。

注意 Oracle Developer Server来自它自己的CD-ROM，简化了服务器的安装和使用。在比较Report Builder和Report Server时，还应该了解Oracle单独出售Oracle Reports产品。

下面介绍使用这些工具创建的各个独立模块的范围和结构。

2.1 Forms

Oracle Developer的Forms组件是开发环境的一部分，可以在其中开发表单模块。它也为开发菜单和PL/SQL客户库模块提供软件框架。框架是一个可重复利用的对象系统，这些对象一起工作，用于定义那些基本的、由特定的应用领域所需求的抽象概念 —— 此处是菜单和表单。用Form Builder建立表单和菜单。

2.1.1 表单模块

表单模块是数据库应用程序的主要组件。在内部结构中，表单是最丰富的模块，它包含许多不同类型的对象。图 2-2显示了组成表单模块的对象的层次结构。其中的编程对象包括了一组对象。在本章的“可重用对象”一节，将会讲述这些对象是什么。

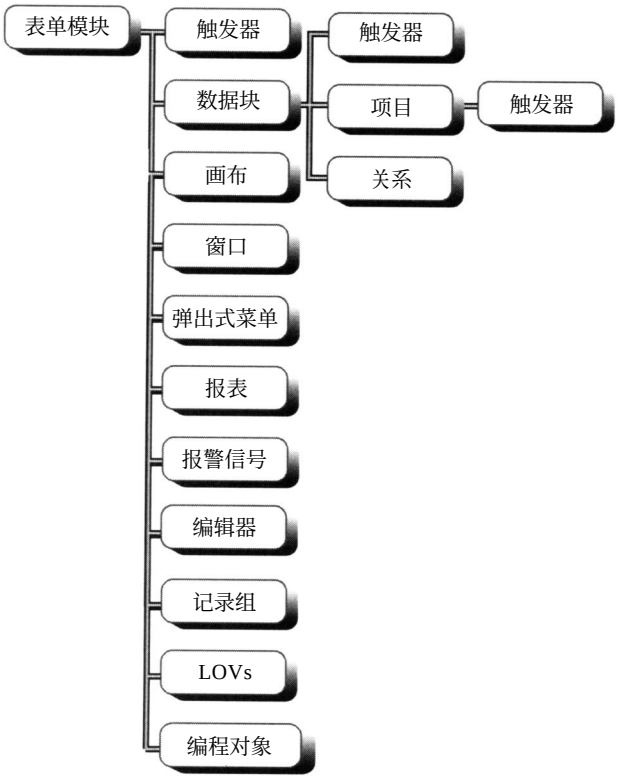


图2-2 表单模块的对象层次结构

China-pub.com 下载

1. 触发器

触发器是一个PL/SQL代码块，可以连接到其他对象的表单、数据块或项目。这些代码表示连接到对象上的业务逻辑。还可以将触发器附到属性类上 (在“可重用对象”一节有更详细的介绍)。当某些事件发生时，触发器“点燃”，执行触发器代码。

在表单应用程序中的触发器包含了加到应用程序中的工作代码，但不包括那些放到独立的程序单元和程序库中的代码。创建触发器及其业务逻辑吸收了表单和代码设计的大部分结果。第6章有详细的活动图，表明不同的表单的结构和触发器是如何触发的。触发器的示例和与其有关的PL/SQL代码将在以后的大部分章节中出现。

所使用的大部分触发器是内部触发器。它是由 Oracle Developer的框架提供的。触发器有若干种，在第5章中有详细介绍。每个触发器有一个专用的名字，如： When-Button-Pressed (当按钮按下时)、Post-Query(登记查询)、On-Delete(删除时)、Key-Help(关键字帮助)、Pre-Update(更新之前)，可以从这些预先定义的名字中选择。如果在不同层次上 (如数据块层或项目层)定义同样的名字，则Oracle Developer执行最低层的那个。例如为数据块中的 20个项目中的2 定义触发器，而同时可以在数据块级定义同名的触发器。对于 2的触发器项目的事件，则执行项目触发器，否则，执行数据块触发器。可以利用改变 execution style(执行方式)属性来改变执行方式。可以在高一级触发器触发低一级的触发器的前后改变执行方式，以取代缺省的特性。可以组合几个触发器的方式来取代执行方式。

可以添加用户自己定义的触发器并直接使用它。这些特点在从前的表单版本中是有用的，现在由于PL/SQL的适应性，就不那么重要了。 Oracle计划在不久的将来取消它。

一个运行中的表单有两个主要的模式：正常模式和“进入查询”模式。在正常模式下可以插入、更新和删除数据库的信息。“进入查询”模式可以创建一个实例记录，用该实例去指定一个对实例数据库数据的查询。 Oracle Developer的这种通过实例查询的方法提供了许多特色，使得在应用中不用代码或者少用代码。

注意 某些触发器不能在“进入查询”模式中使用(参看“数据块”部分)，因为它们的活动和数据操作有关，因此在这种模式下无意义。

某种触发器，如按键触发器，是在用户界面的组合按钮按下时启动。添加按键触发器允许对键盘重新编程，使按键的动作与标准键盘的定义不相同。因为这些触发器是非标准的，在标准菜单的帮助栏中，触发器定义的内容可显示出对那些被列出的可用键的描述。

2. 数据块

数据块是表单的中间建造单元。可以用两种方法考虑数据块：作为数据项目的集合或者作为记录的集合。每个记录的结构是相同的。限定每一次显示的记录数目，而且指定用水平或垂直方式显示记录。

有两种数据块。“基于表的数据块”对应于数据库中的一个关系表、一个对象表或一个视图，管理某些数目的记录，这些记录对应于表或视图中的相应的行。基于表的数据块还可以表示一个存储的过程。另一种数据块是“控制块”，它不与表、视图和存储过程相对应。通常，控制块表示一些单值项目目的集合，实际上只有一个记录。例如，需要跟踪一组记录的合计值、总计值或平均值，那么可在控制块中创建一项目来表示那个值。虽然对于控制块来说，那个项目只有一个值，但是，却相当于基于表的数据块的一组记录。

数据块的显示有两个特殊的特性与作为记录的容器有关。数据块可以有一个滚动条，让

China-pub.com
下载

用户管理那些显示内容大于显示屏幕区的记录组。它还有一组专门的可视化属性，在本章的“可重用对象”中讲述。这些属性为当前记录定义不同的显示，当前记录是指光标当前停留的记录。

通常，在数据块内的“导航”按照记录定义数据项目的次序进行。运行时，系统具有从记录移至记录、从项目移至项目、从数据块移至数据块的功能。当从记录的最后一项目出来时，通常返回到同一记录的第一项目。Form Builder允许重新定义这个缺省的属性。可以允许跳到下一个记录(如果从第一项后退的话，即到前一个记录)，也可以允许从最后一项或第一项移出时，跳到下一个或前一个数据块。可以设置所指定的数据块为下一块或前一块，创建一个数据块连接表。还可以告诉表单在数据块菜单中输入数据块的名字，这样，可以通过在连接表中选择数据块的名字的方式跳到被选择的数据块中。

基于表的数据块的主要功能是为数据库的关系表或对象表或存储过程提供一个接口。Oracle Developer提供一个Data Block Wizard，帮助从数据库模式(schema)中建立这些数据块。向导还帮助从数据库模式建立主-从(master-detail)数据块。这些数据块按照父子关系建立关系，比如发货单及其所属项目。

Oracle Developer通过构造基于数据块及其结构的SQL语句来自动地管理数据。因此，数据块具有若干管理SQL语句结构的属性：

- Enforce Primary Key and Key Mode(强制执行主关键字和关键字模式)：Enforce Primary Key特性缺省地，Forms使用Oracle唯一的内部标识符——行标识符rowid——来标识在SQL语句中产生的行。Enforce Primary Key特性告诉Forms，改用项目及其值来标识行，该行与数据库表中组成主关键字的列所相对应。Key Mode属性还告诉Forms UPDATA语句中是否包含主关键字值。有些数据库管理器不允许更新主关键字值。
- <operation>Allowed<操作>允许：可以对数据块禁止使用指定的SQL语句——SELECT、INSERT、UPDATE、DELETE。例如，为了防止用户删除行，可以设置Delete Allowed为No。
- WHERE子句：这是SQL的一部分。Forms将其加到为查询而产生的SELECT语句上。有关细节请参看第5章的有关内容。
- ORDER BY子句：这也是SQL的一部分。Forms将其作为查询而产生的SELECT语句的ORDER BY子句。
- Optimizer Hint(优化提示)：Forms将这个Oracle SQL组件加到SELECT语句中，使用Oracle的专门的优化特性加速查询。
- Update Changed Columns Only (仅更新变化了的列)：如果这个属性设置为Yes，那么Forms只将变化过的列包含在它产生的UPDATE语句中。否则，不管这些值是否变化，将更新所有的列。通常应该将其设置为No。
- Locking Mode(加锁模式)：在改变某一行的值时，这个属性确定是在提交数据库的变化时锁定该行，还是立即锁定这一行。
- Data Source and DML Target(数据源和DML目标)：这个属性用来确定哪种数据源给数据块提供数据，或者向哪里写数据用FROM子句、表名字、存储过程或者事务处理触发器。
- DML Array Size (DML数组的大小)：这个属性确定一次插入、更新或删除的记录数。

China-pub.com 下载

数据块还为表单的query-by-example(用示例查询)属性提供接口。用户可以将数据块放在Enter Query模式中，然后在示例记录的域中输入查询判据。当这个查询被执行时，表单根据其值自动建立SQL SELECT语句。结合使用WHERE子句和ORDER BY子句的特性，可得到综合的查询条件。在Enter Query模式中，还可以为WHERE子句写入自己附加的SQL。可以指定存储器缓冲区记录数、最大查询时间和一次取入表单的最大记录数。

3. 项(项目)

项(项目)是表单最主要的构造单元 (在本书中，为了表达清楚起见，有时使用“项目”这个词。“项”和“项目”是指同一种东西 —— 译者注)。项有许多属性 —— 多得难以在此概括。本节只提供项的功能摘要，而不涉及属性的细节。使用专用的在圆点分隔的语法来使用项：<data block>.<name>。在这里，<data block>是拥有项的数据块的名字，<name>是项的名字。在数据块中，项的名字是唯一的。

表2-1描述了不同类型的项目。说明中提到的任何特殊的属性只与特殊类型的项目有关。

表2-1 Forms中项目类型及其用途

项目的类型	说 明
ActiveX控件	显示ActiveX控件，参看第15章
Bean域	显示与表单结合的java bean的输出，参看15章
图表项目	显示来自图形形式模块的图表
复选框	让用户接通或断开布尔值
显示项	显示数据的“Read only”(只读)文本项，不允许用户改变
分层树	显示类似于Windows Explorer或对Object Navigator的树结构通过树操作数据
图像	显示一个位图或矢量图
列表项目	若干正文行的列表，可以作为弹出式列表、固定尺寸列表或组合框显示
OLE容器	显示OLE对象，参看第5章
按钮	使用户执行一个与按钮有关的触发器 (当按钮被按下时)
单选按钮组(单选)	在若干按钮中接通一个按钮，其他所有的按钮断开。这个对象拥有一组无线按钮，这些按钮一起工作，每个按钮表示一个特殊值，它表示数据库的一列
声音	显示一个表示声音的图标，单击图标则有播放
文本项	在单行或多行的域里显示文本
用户区	显示任何用程序装入这个空间的合适的内容
VBX控件	显示VBX(16 位)控件

每一个特定的类型有一组特性。例如，文本项可能有滚动条，单选按钮可能有 Other Values(其他值)属性(该属性规定对于非单选按钮组中的值，数据库查询将发生什么的事情)。因为项目是数据值，所以项目有若干和数据有关的属性。表 2-2列出了Developer表单正常的不同的数据类型。某些类型的存在是为了和以前的表单版本兼容，则未列入表 2-2中(例如EDATE、JDATE、RMONEY，等等)。如果可能，应该避免使用这些类型，而只使用表 2-2中的类型，以保证所有的项目能够直接正确对应到 Oracle 的数据类型。

表2-2 Forms中项目的数据类型

数据类型	说 明	避免使用/可以使用
Alpha	任何字母符号—大写、小写或大小写混合	避免使用
Char	任何字母或数字符号(与Oracle的VARCHAR2对应)	可以使用
Date	有效的日期和时间(与Oracle的DATE类型对应)	可以使用
DataTime	有效的日期和时间(与Oracle的DATE类型对应)	可以使用

China-pub.com
下载

(续)

数据类型	说　明	避免使用/可以使用
Int	任何带符号或不带符号的整数值	避免使用
Long	扩展长度的字符串	可以使用
Number	带符号或不带符号的定点数或浮点数，用或不用指数/科学表示法(与Oracle的NUMBER列对应)	可以使用

有若干用于控制项目值的格式和验证的属性：

- Format mask (格式掩码)：一个广泛的各种值的说明，格式化输出或者验证输入。
- Default value (缺省值)：在创建项目时分配给它的值。
- Copy value (复制值)：在创建项目时，由其他项目复制过来的值。
- Length and Range (长度和范围)：数据的最大长度和范围。
- Must Enter (必须输入)：确定用户是否必须输入值的设置。
- LOV ：一个与项目有关的列表值。使得输入的值是一组指定的值 (参看“记录组和LOV”部分有关的内容)
- Editor(编辑器)：正文编辑器对象。用来输入正文。

项目具有全部可视属性：字体、颜色、填充、斜面等等。关键的属性是是否显示项目。可以有一个在任何地方都不出现的项目，但是它保持有值。可以利用这些项目作为 PL/SQL中的代码的变量。一个特殊的用法是从数据库取值送入这些项目，然后根据这些值计算出其他的显示值。还可以书写工具提示，当鼠标移至某项目时，系统显示所写的提示。

项目的另外一个关键的显示特性是显示项目的画布的名称 (参看“画布和窗口”部分的内容)。通过改变画布的名称，可以完全控制项目在显示中什么时候出现和在什么位置出现。还有一个tab(标签)特性，可以使项目在标签画布视窗上按照特殊的标签移动位置。 将在后面看到有关的内容。

每个项目有一个提示符，即在画布上紧靠着项目显示的一个字符串。提示符的属性包括位置(在项目的左面、右面、上面或下面)、与项目的距离和提示符的显示特性 (字体、颜色等等)。当项目移动时，提示符也在画布上在项目的周围跟随项目移动。所以不必操心在画布上建立样板文本(boilerplate text)，也不必担心要在画布上重新排列项目时的管理问题。

有可能不允许“导航”到项目内，可能会完全禁止，或者只允许通过单击鼠标“导航”而不允许用户从项目移动到项目。可以把一个项目与另一个项目相连接，以强制导航按照与其他项目定义次序不同次序进行。

一系列属性影响Oracle Developer Forms在建立SQL语句中使用项目的方法。最重要的一个属性是项目是否与数据库的基础表的列对应。其他的属性控制查询的性能、数据操作的性能和锁定性能。

Implementation Class的属性可以超越Java Bean的控制特性。可以为项目的类型指定实施类(Java bean类的名称)。范围从Bean area到复选框、列表项目、按钮、单选按钮组或文本项目。有关Java Bean控制的详细示例请参看第15章。

最后，有个叫做calculation(计算)的组。它包含若干属性，可以通过公式或汇总其他项目的值将项目转换到计算域，然后，可以创建总计域、汇总域或者所有感兴趣的计算域。可以自动维护这些域，不用写任何触发器。有关计算项目的示例参看第 6章。

4. 关系

China-pub.com 下载

关系是一个对象。Oracle Developer的表单用它来构造主-从(master-detail)表单。属于主数据块的关系对象表示主记录与从记录 (detail records)之间的关系。关系的主要属性是从记录数据块的名字和表单管理关系所用的连接条件。

可以特指一些特殊的有关删除主记录的特性 (不管是否删除从记录)或当没有主记录时，插入或更新从记录。第6章讨论这些属性和与从记录的自动显示相关的设置。

5. 画布和窗口

画布是放置文本和项目的背景。每个项目准确地引用属性工作表中的一个画布。可以将一个数据块分到各个不同的画布上。就是说，数据块可以在不同的画布上显示它的项目，而不只是在单个的画布上显示。

画布并不是独立的界面对象。要查看画布及其他的项目，必须在窗口中显示画布。窗口是应用程序的一个矩形显示区，被一个框架所包围，通过 GUI 平台维护。Oracle Developer建造这些各自独立的对象，使用户可以构造窗口，从而提供一个画布的视图。这个视图是窗口内的一个矩形，覆盖整个画布或覆盖部分画布。可以通过窗口查看的那一部分，就是视图。窗口有垂直或水平的滚动条，可以在画布的周围滚动查看不同的视图。

画布有五种类型：

- Content (内容)：显示窗口的基本内容。
- Tabbed(标签的)：用一系列的带有标号标签的画布显示窗口的基本内容。
- Stacked(层叠的)：在许多的其他画布上显示那些有条件的或者可以分离的内容。可以规定在显示的窗口中是否立即显示层叠的画布，或者是使画布处于不可见，而在需要时才使它可见。
- Vertical toolbar (垂直工具栏)：将工具图标的内容显示固定在窗口左边的垂直工具栏内。窗口通过名字来指定在工具栏中使用的画布。
- Horizontal toolbar (水平工具栏) ：将工具图标的内容显示固定在窗口顶部的水平工具栏内。窗口通过名字来指定在工具栏中使用的画布。

不仅可以在画布周围滚动窗口，而且可以通过层叠画布在同一个窗口中显示多个画布。每一个层叠画布可以有它自己的滚动条，使得能够在其他画布不动的情况下滚动某一个画布。有关如何使用这个特性的细节，请参看第 6 章“层叠画布”的内容。

可以定义一个画布中具有多个标签页。这使得能够建立标签对话框，在对话框中具有多个视图，可以在标签页上单击访问这些视图，如图2-3所示。

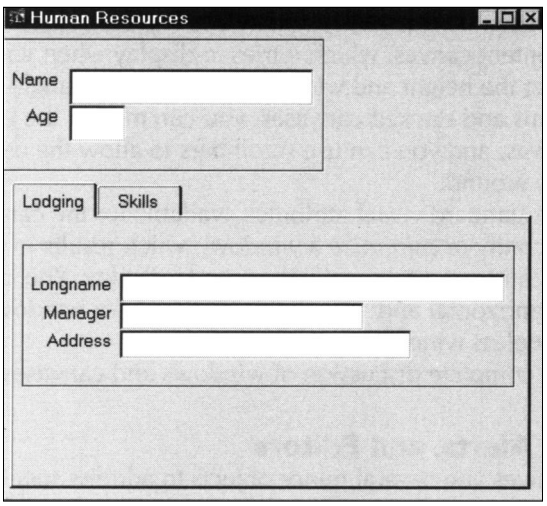


图2-3 多个标签页

标签页画布视图用一种清晰而方便的方法将大量的项目组织到单个的窗口中。通过Layout Wizard建立这些是很容易的。Layout Wizard可以引导你建立和修改所建立的数据块的画布布局。

窗口可以是模式的，也可以是非模式的。如果是模式窗口，则用户应用程序在其他窗

China-pub.com
下载

口中做任何事情之前要先解除这个窗口。非模式窗口可以使用户不用解除当前窗口就可以移到另一个窗口。通常，模式窗口有一些特殊的属性，如没有滚动条、窗口大小是固定的、无法使图标最小化。

当“导航”到另外的窗口的项目的时候，可以告诉 Developer将非模式窗口隐藏起来。但是，Developer不在模式期间定义两个窗口的类型，而是用这样的方法定义：

- 文档：是一个通用的非模式窗口，在其中显示一个应用程序的“文档”对象，通常是与应用程序主要部分有关的内容的画布。
- 对话框：一个通用的模式窗口，在其中显示一些选择项目或者其他的控制应用程序操作的方法。

按照性质，对话框通常是模式的。尽管在应用程序中做其他事情的时候，可以使非模式的对话框继续存在。

画布对象指定显示窗口的名字，建立画布与窗口之间的关系。窗口对象通过名字指定其主要内容的画布，即在第一次打开窗口时显示的画布。可以设置画布的高度和宽度。如果想要显示不同的项目和层叠画布，则可能要修改窗口和画布的高度和宽度。可以使用滚动条让用户移动画布和窗口。

可以使画布和窗口具有所有标准的可视属性。可以使窗口图标化和最小化，使窗口当作图标那样显示，可以指定要显示的图标。还可以指定窗口中显示垂直滚动条还是水平滚动条(通常，这些只对非模式的窗口有用)。

在第6章详细讨论窗口和画布。

6. 弹出菜单、警报和编辑器

Oracle Developer提供了几个次要对象以满足特殊需要。第一个是弹出式菜单。弹出式菜单是一个浮动菜单，当在画布或项目上单击鼠标右键时，弹出一个菜单。弹出菜单是在Windows中较新的改进，但是在其他的窗口系统中已经有较长的历史了。这些菜单让你用一种比较自然方便的方法设置对象的行为动作。让用户很快“熄灭”应用程序中某些与鼠标位置有关的活动。

警报是特殊的对话框。它用图标和一个至三个按钮来显示信息，例如“OK”和“Cancel”、“Yes”和“No”，或者其他的按钮。还可以规定三个按钮中的某一个按钮作为缺省按钮。可以用画布将其作为一个窗口来建立。但是使用警报对象可以不经过那些忙乱而很快将对话框放在一起。有三种类型的警报：Stop(停止)、Caution(小心)和Note(注意)。应用程序中显示的简单消息就是这样分类的。

编辑器是一种简单的对话框，其作用是让你将文本行输入到文本项目中。编辑器对象让你指定窗口的大小、可视化属性、编辑标题和窗口的其他属性。让你用不同的方法对不同的文本域创建不同的编辑器。

7. 记录组和LOV

记录组是一种特殊的数据结构，它与表类似，具有行和列。记录组可以是查询记录组，或者是静态记录组。用SQL SELECT语句定义查询记录组。查询的SELECT列表给出这个记录组的列结构。通过在专门的对话框中输入的列说明来定义静态记录组。有几个内部子程序在运行时操作记录组。可以为 LOV使用记录组(参看后面的内容和第 4、6章)。要传递记录到报表中或图形中去作为显示数据参数时或作为 PL/SQL数据结构时，使用记录组(参见第15章)。

China-pub.com 下载

提示 记录组可以从Database Server(数据库服务器)中检索数据到用户存储器，然后可以像引用数据库的表一样使用这些数据。但是，由于数据库表在用户的存储器中，就使得访问非常快而且减少了网络的信息流量。可以用这种方法，用记录组来优化应用程序。

LOV(list of value，值列表)是特殊的对话框。它显示一个记录组并且让你选择其中的一行，并返回单个值。可以将 LOV作为一种方法来使用，即从专门的一组值中进行选择，例如，LOV和文本项目结合提供了比较容易输入值的方法，并且提供一个值的列表。对照这个列表可以确定用户的数据输入。自动选择特性可以使终端用户进入字符串方式来自动搜索列表中的项目。还有，可以使用标准的窗口和画布来建立这个对话框。 LOV对象给出一个捷径和方法，在域中自动得到LOV特性而不用任何编程。LOV Wizard简化了整个过程，并且可以使用Query Builder来建立对LOV值的SQL查询。第4章讲述在表单中如何使用LOV。

8. 可重用对象

在“可重用对象”的标题下，这部分概括了几个用来构造应用程序组件的对象。这些对象同样适用于表单和菜单模块，其中的某些 (附加的库和程序单元)也适用于库模块(见图2-4)。

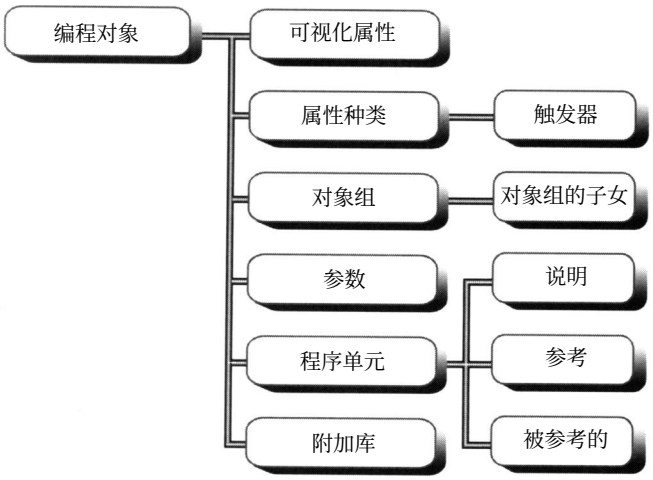


图2-4 构造应用程序部件的对象

被命名的可视属性是若干个显示属性的集合，比如字体或者加到对象 (项目、画布、数据块)上的背景色和前景色。如果从其他的对象中引用这些可视属性，那么引用的那些可视属性将取代这个对象的属性。如果改变可视属性的特性，那么在所有的对象中的属性将继承可视属性的特性。

属性类是一组对象属性(任何一个对象的属性)，目的是附加于一个对象。就像可视属性一样，当一个属性类附加于对象时，可以使得该属性类的所有属性对于所定义的对象均有意义。还有，当改变属性类时，连接了该属性类的对象可以继承或者也可以不继承这些新值，这取决于这些属性值在对象层上是否是被取代。

可以从属性类中创建子类，或者从其他对象划分子类。这种特点可以使得在对象中增加新属性或新方法，或者改变已有的属性和方法。

对象组使你包装可重用对象，为以后的复制或划分子类所用。对象组在单个标题下的模块中收集一组对象。通过复制或者划分对象组的子类，可以得到它包含的所有对象。可以将任意的对象向下组合到数据块层，但不能对块内的项目进行分组，必须在对象组中包含完整

China-pub.com
下载

的数据块。

对象库是一个模块。通过拖动对象并放入到库模块中的方式，集中可重用的对象集。要重新使用对象库中的对象，就将库打开并将对象拖动并放入应用程序中。利用库可以将对象组织在一个或多个标记文件夹中。一个对象库可以包含单个的对象，如块、窗口、警报、属性类等等，也包含对象组。

第10章解释命名的可视属性、属性类、对象组和对象库的细节，还说明如何有效地使用它们来建立可重用的组件，以及如何将标准的属性组放入应用程序中。

参数是在模块层上定义的表单、菜单、报表或显示的数据对象。在模块的范围内可以利用参数作为任何PL/SQL程序的变量。参数有数据类型、缺省的初始值和最大长度。可以在启动表单时设置参数。设置方式有：通过传递一个值；通过将参数放到命令行中 (PARAM=“VALUE”)；通过将参数传递到参数清单中(参看第5章)。

程序单元是在模块范围内定义的PL/SQL程序包、过程和函数。所有的Developer模块允许将程序单元定义为模块的一部分。库模块有点特殊：它将程序单元组装为模块，并且通过连接的库对象，使其可以为其他的模块使用。这是对已经定义为模块的库的引用可以调用在模块中的由模块的PL/SQL定义的程序单元。不能调用在没有连接到的其他模块或库中定义的任何程序单元。

每个子程序的程序单元都要显示它的技术要求 (过程或函数的参数或函数返回的类型)。每个程序包的程序单元要指出它的子程序 (程序包子程序)。每个程序单元还要说明这个程序调用什么程序单元(引用对象)和哪个程序单元引用当前这个单元 (被对象引用)。第12章解释程序单元(程序包、过程和函数)的细节。

2.1.2 菜单模块

菜单模块比表单模块要简单得多。如图2-5所示。

图2-5 菜单的组成

菜单模块由一组编程对象(和前述部分所讲述的相同)和一组菜单组成。每个菜单又由一组项目组成。第6章详细地讲解如何定制菜单。

菜单项目可以是一些简单的项目，或者可以是几个特殊的菜单项目形式：

- Check(复选)：旁边带有一个复选标记的菜单项目。这些项目使得通过菜单而不是通过对话框允许使用或者禁止使用所选项目。
- Radio(单选)：是一组互不相容的菜单选项。如果选择了一个，就不能选择其他的项目。
- Separator(分隔符)：这个菜单项目不做任何事情，只是分隔其他的项目。通常是菜单项目上的空间或空行。
- Magic：平台专用的菜单项目，如Cut(剪切)、Copy(复制)、Paste(粘贴)、Undo(撤消)和Help(帮助)。

  每个菜单项目有一个命令。当选择该菜单项目时就执行这个命令。可以是这几件事情之一：

- Null(空)：不做任何事情。分隔符必须是一个空命令。
- Menu(菜单)：显示一个子菜单。

China-pub.com 下载

■ PL/SQL：菜单执行一个PL/SQL块(包括Run_Product和Host)来执行其他的Developer程序或者操作系统命令。

也可以将菜单与一个或多个安全角色联系起来。第 10章讲述安全性和菜单的详细内容。

2.2 库模块、内置软件包和数据库对象

Developer中的几个对象出现在所有的组件中：程序库模块、内部程序包和数据库对象。

程序库模块与其他模块比较起来是相当简单的。只有一个程序单元和一组附加的程序库组成。有关程序库和所包含的程序单元的细节，请参看第12章。可以在文件系统或数据库中存储库，就像存储任何模块一样。图2-6是库模块的结构。

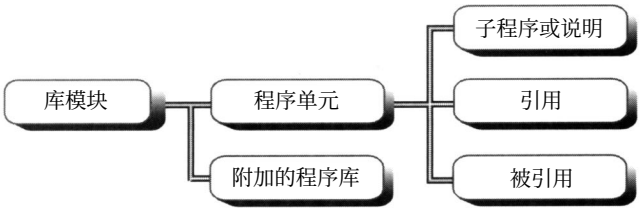


图2-6 库模块

所有的Developer生成器访问一系列内置的PL/SQL程序包，然而程序包不同于其他产品。这些程序包提供了一组范围很宽的工具来操作模块和组件中的其他对象 (Forms、Reports或Graphics)。每个程序包都有技术说明，列出程序包中的子程序以及其说明(参数及函数返回)。每个程序包在产品参考手册中有多方面的文档资料。应该查阅这些文档资料，查看程序包的类型和异常事件以及使用不同的子程序的细节。

还有，所有的生成器都可以访问连接的数据库的对象。图 2-7表明了数据库对象的对象分层结构。生成器列出数据库的所有用户。对于每个用户来说，可以看到可访问的存储的程序单元、库、表和视图。用户只能查看那些通过注册的用户名能够访问的对象。例如，如果是作为没有特权使用特殊表的用户，就不能在数据库对象中查看到在这个数据库对象中列出的表。库是存储在数据库中的库模块，与文件系统完全相反。

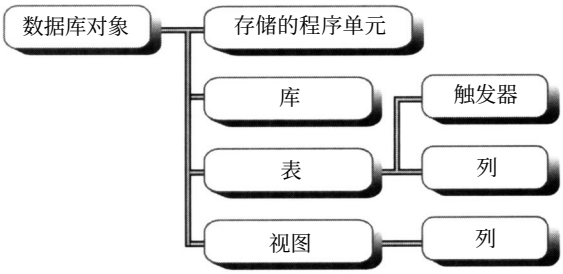


图2-7 数据库对象的分层结构

表也显示了表的数据库触发器。这些触发器是一些 PL/SQL块，在数据库中有特定的事件发生时执行它们，比如INSERT或DELETE。虽然与Developer代码的触发器类似，但是数据库触发器响应数据库事件而不响应应用程序事件。表和视图也显示它们的列以及数据类型。

可以通过从其他的模块拖动程序单元到存储程序单元的数据库对象区域创建存储程序单元。这样可以将子程序或程序包的处理移入数据库，允许开发多层体系结构。通过用这种方法移动代码常常可以改善性能或可靠性。

2.3 Reports

Developer的Reports组件是开发环境的一部分，报表模块是在开发环境中开发的。而且在这个环境中可以引用外部的查询对象，可以设置和存储调试对象。 Report Builder也包含库和

China-pub.com
下载

数据库对象。

报表模块有复杂的结构，如图 2-8所示。报表的基本组件是它的数据模型、参数形式、报表触发器和布局。第 7章讲述报表的不同特性。它也可以有程序单元和附加程序库，就像在本章“可重用对象”部分中讲述的那样。

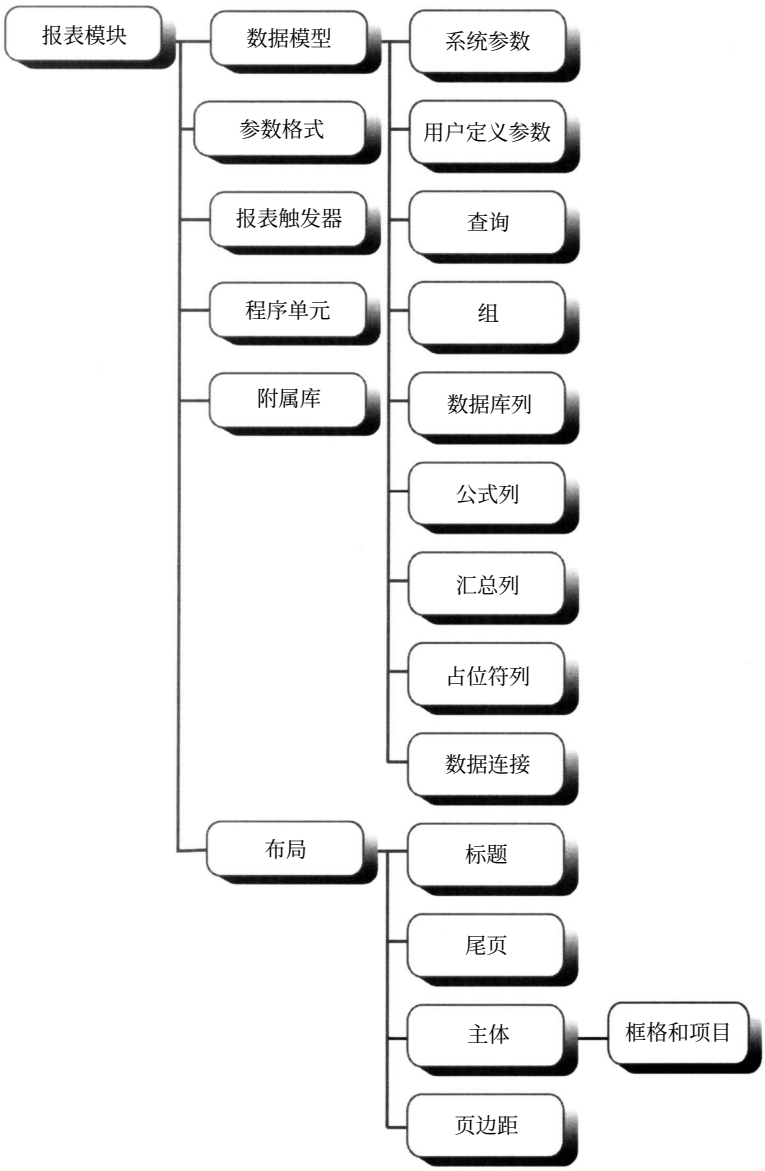


图2-8 报表模块的对象层次结构

报表的“数据模型”是数据结构及其在报表中不同的表现。可以在专门的图形编辑器中创建数据模型。

参数对象是模块变量，可以在 PL/SQL代码中和任何接受数据值的地方引用它们，就像输入的数据一样。“系统参数”是Reports自动定义的参数，“用户定义参数”是用户定义的参数。设置运行时的参数有缺省格式，或者可以在报表中建立完全不同的格式作为参数格式对象。

China-pub.com 下载

当运行报表，提示用户设置指定的参数的时候，报表程序运行这个格式。第 10章详细讨论报表参数、缺省参数格式、用户定义参数格式和利用参数使报表可重用的所有方法。

查询是一个SQL语句，它返回数据值，这些数据值是报表的基础。可以将查询嵌入报表，或者使用“外部查询对象”：即在独立文件中的简单 SQL文本，可以由应用程序共享。“组”将查询返回的记录标记为报表中的重复组。组的层次结构表示记录之间的嵌套。“数据库的列”是来自查询的SELECT列表的列。“公式列”是用PL/SQL块计算的特殊列。“汇总列”是报表中指定用来为多个记录存放汇总信息的列，就像小计或总计那样。这些列在高于所概括的记录的层次上编组。“占位符列”定义用来从触发器、公式或用户退出时填入的列，而不是用数据或标准的汇总填入的列。例如，可以导出专门的报告域，报告在一组地域行中那些销售最多的地区的数据。“数据连接”是用来在主-从(master-detail)报表中将一组行连接到另一组行。

报表的“布局”是它的图形结构。“标题页”在前面，接着是“主体页”，最后是“尾页”。报表的“页边距”是指在标题页、尾页和文本页的正文边界以外的区域。报表主体包含全部重复的框格和报表项目，用图式结构将数据模型放到格式化的报表之中。标题和尾页包括任何数据元素或者样板，分别使得前边和后面的报表体有意义。可以通过 Live Preview(live previewer)操作这些对象中的大多数，可以在报表形式的友好的用户界面上隐藏对象零乱的细节。

表2-3概括了几种报表的布局，每一种布局可能对伴随的数据模型有特殊的要求。

表2-3 Developer的Report Layout

Report Layout的类型	说 明
表格(Tabular)	缺省类型。这种结构是在每一页重复用列的标题组织一系列的数据行
主-从(Master-Detail)	将两个或多个组构成一些关联的、嵌套的行，目的是对应于外部的组的每一行，显示一个行的集合
表单(Form)	象表单一样组织数据，用一行格式化后的数据放在若干排上，个别的域的左边有标记
格式信签(Form Letter)	是样板文本和数据域的混合体。它用数据模型中的数据行代替样本文本，目的是产生同一表单的多个拷贝，每行一个
邮件标签(Mailing Label)	在每一页的固定尺寸的框中打印字段的重复组
矩阵(Matrix)	称为交叉表的报表更合适。它用有标题的行列网格显示数据

第4章介绍如何创建数据模型和报表布局。第 7章更详细地继续介绍有关的内容。每种布局有一些选项，可以使用相对来说少的基本布局做出数目惊人的事情。 Report Wizard帮助你通过一系列交互对话建立报表。

“报表触发器”是一些 PL/SQL代码块，这些代码在含义明确的“点”上执行：在报表之前；在报表之后；在页面之间；在参数表单和参数表单之后。在报表中也有其他的触发器，但是不独立，用在表单中的方法命名对象。通过拥有它们的项目来访问它们。第 5章详细讲述报表触发器的触发和如何书写它们的说明。

Report Builder还将调试信息结合到可以利用的对象中，可以创建调试活动 (可以在PL/SQL代码中设置断点)。调试活动在调试会话期中持续。它还包含一个在运行时查看的有关调用堆栈的。有关调试报表和在 PL/SQL调试程序中有关这些对象的细节，请参看第 13章。

2.4 Graphics

Developer的Graphics组件是开发显示模块的环境的一部分。显示模块是从数据库的数据

China-pub.com
下载

导出的一个或多个图表，它可能包含图形元素的任意组合，其中包含或不包含对数据库的引用。可以利用显示模块精确地显示业务数据的图形，或者将它们当作画图工具，或者两者兼用。

如图2-9所示，在对象的项目中的图形显示的结构是简单的。显示模块包括布局、模板集、查询、参数、声音、计时器，以及与其他模块所包含的相同的程序单元和附属库对象。

图2-9 图形显示模块的对象层次结构

布局包含了在层次结构中显示的图形组件，在层次结构中表示了它们的关系。它还将这些元素与查询的列联系起来。可以作为图形产品的一部分而得到几个特殊类型的图表，如表 2-4所示。

“图表模板”是定制的一组选择项目，可以采用相同的格式在同一个显示中创建多个图表。例如，可以用一个模板建立一系列的饼图。第10章将详细讲述怎样创建可重用图表定义的模板。

查询是一个SQL SELECT语句，与在报表模块中的一样，它定义一个用图表显示的数据集。构造查询是用来返回那些用精确格式显示的所想要的图表的数据。例如，饼图不能概括各个部分的数据，而必须用 GROUP BY和SUM功能来完成这个工作。第4章讲述如何建立这种查询。

参数，如同报表一样，是模块的变量，它们在 PL/SQL代码模块中到处定义和使用。参数可以在命令行中设置，或者可以通过从其他的程序如 Forms或Reports中传递参数列表来设置。第10章中有详细的内容。

除了可视化组件外，还可以将显示与声音对象结合，以创建完美的多媒体显示。

专用的计时器对象在图形显示中作为一种报警时钟。可以规定一个时间段，当该时间段结束时，计时器便执行某些PL/SQL代码。用计时器可以实现各种特殊的效果。

表2-4 Graphics Display的图表类型

图表类型	说明
列	用垂直条显示数据组
饼	将数据组显示为圆饼的各个部分
条	用水平条显示数据组
表	用表格形式显示数据值
线	用与X值和Y值有关的线段显示数据值
散列点	用与X值和Y值有关的点显示数据值
混合	用与X值和Y值有关的线段和点显示数据值
高/低	数据元素用三个值的组合显示：高、低和结束。例如可以显示股市的价格
双Y	对每个X轴的值显示两个Y值，例如可以显示随时间变化的两个变量
甘特图	显示水平条来表示开始值和结束值，通常表示进度表。用 X轴表示时间，Y轴表示进度表中的一组任务

Graphics Builder与Report Builder相同，也综合相同的调试信息。可以创建调试活动——可以在PL/SQL代码中设置断点，这些断点在调试会话期间将持续存在。它还包含一个在运行时供查看的堆栈调度显示。第 14章详细讲述如何调试图形显示和如何在 PL/SQL调试程序中使

China-pub.com 下载

用这些对象。

2.5 螺母和螺栓

对于应用程序来说，每个对象都有强大的功用。但是， Developer比它所包含的那些对象的作用更大。Developer是一个系统，它将所有的对象组合在一起，成为一个工作的整体。不仅是这些对象在一起工作，还可能用其他的应用程序的对象工作，以及用其他不同于 Oracle的数据库管理系统的数据工作。

Developer的主要黏合剂是PL/SQL。这种编程语言在触发器、程序单元和其他的程序对象中使用。通过访问项目和参数 (或者甚至是全局数据)，PL/SQL导致发生与所有的对象绑定的事件。利用Developer的组件中的内部子程序，可以操作其他组件中的对象：

- 在报表和表单中嵌入图形显示。
- 传递报表或表单的数据到图形。
- 从图形显示或表单中产生报表。
- 从图形显示中将参数值读入表单。

在各个组件中的专门的子程序使得可以与标准 Oracle以外以及Developer环境以外的事件进行交互。可以使用 Host过程通过操作系统命令接口调用其他程序。在 Windows平台上，可以使用OLE2，ActiveX控件和DDE来操作其他应用程序的对象。还可以利用 Open Client Adapter访问Oracle以外的数据库管理系统，对于这些数据库管理程序可以使用 ODBC驱动程序。可以用这些方法在设计环境和运行环境中完全代替 Oracle。或者，还可以在Oracle下开发而在运行环境中却使用另一个数据库管理程序。例如，用户可以将他们的数据保存在 SQL Server、Sybase或Informix等数据库中。

所有这些选择使得Developer成为一个结构好、健壮的和向 Internet应用程序开放的坚实有力的环境。本书的第二部分和第三部分详细介绍如何原型化 (prototype)、设计、编码、测试、调试和部署终端用户的应用程序。第四部分是可以用于 Developer应用程序的所有各种对象的综合参考信息。

China-pub.com
下载

第二部分　Oracle Developer教程

第3章　数据库、表单和报表共同工作

本章为下面几章的教程内容作准备。它建立教程使用的数据库，然后介绍在 Oracle Developer中建立表单、报表和图形的一些规则。

3.1　建立Talbot数据库及其应用程序

在我们创建数据库以前，让我们回顾一下历史， Talbot农场是进入新世纪时在新罕布什尔州的一个工作农场。这个农场位于靠近Keene镇和Edmeston镇的新罕布什尔州西南的乡村地区。以前的一本书(George Koch和Kevin Loney著《Oracle 8完全参考手册》，曾经使用Talbot农场作为它的例子的一部分。该例子非常适合 Oracle Developer教程的需要。

这个例子出自被George Koch发现的一个旧的总帐帐簿，上面有从 1896年开始直到1905年的记录(图3-1)，这个帐簿是Dora Talbot农场的普通帐簿，可能是由他的儿子 George B.Talbot保存。

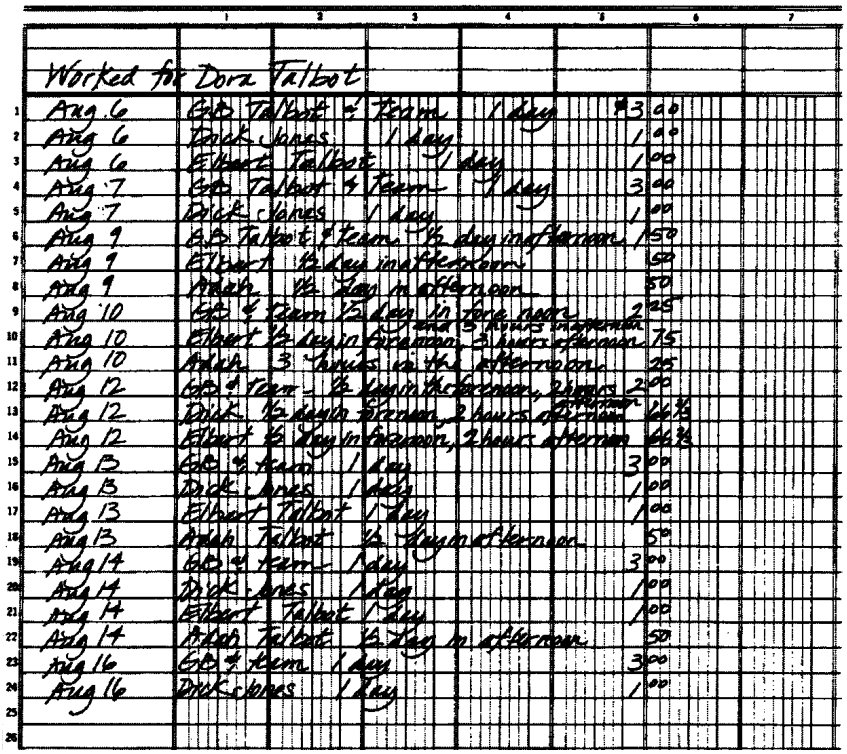


图3-1　Talbot帐簿的样本页

China-pub.com 下载

这个Ledger簿包括销售、购入、劳动者的薪金支付和材料的采购：所有的按照顺序的工作业务的事务。它还包含有另外一些有用的信息，如在农场工作的临时工的地址清单。

这个帐簿启发George Koch把它作为在关系数据库之前人们是如何记帐的一个好例子——而且作为在旧式的帐簿记录和现在的关系数据库的记录之间的相似性的有力说明。他使用这个帐簿作为一个小的但是极重要的关系数据库的例子。这个帐簿可以很容易变成一个关系表。最初的数据库为如何写 SQL提供了一个相关的例子。这本书进一步发展了这个想法，并将Talbot帐簿用于Oracle Developer开发应用程序。

《Oracle 8完全参考手册》实际上并不与开发应用程序的任务有关，而只是和 SQL和Oracle的教学有关。那本书的例子虽然正确有用，但未坚持那本书提倡的过程和设计建议。在这里，你将看到一个更完全的、带有全部完整的约束的 Talbot数据库版本，它将作为在这本书中采用的综合例子。

这本书还自由地扩展了过时的事物，使某些 Talbot农场范围之外的东西进入农场的范围，为了现代数据处理的需要而扩展，将其变成扩展了的现代商业。 Talbot的操作可能是使用牛和雇员去挖坟墓或打马掌，但是这本书却要面向下个世纪——无论是什么样的世纪，要把它作为是现代共同农场的一个例子。 Talbot农场将增长，而且它的顾客和供应品遍及全世界。对全球的问题要求全球的方法，故将 Talbot扩展到访问因特网。因此，请暂停对历史的怀疑，这不过是用Talbot的例子说明如何在自己的业务中有效地使用 Oracle Developer而已。如果有运气，它将继续存在(只是至少要保藏一个Oracle Developer应用程序，为了在2095年被发现用来作为器官数据库或量子数据库的例子，或者作为未来才有的无论什么技术的例子)。

3.1.1 Talbot的需求

任何数据库应用程序的工程均从应用程序系统的用户要求开始。 Talbot的要求集中在两类查询和维护功能上：总帐以及在 Talbot工作的人员的名单，其中名单包括他们的地址和技能。

注意 在现实世界中，有许多需求，但可能有的需求超出了范围，需求的内部定义和外部定义超出了本书中教程和参考资料的范围，本章的材料提供了在现实系统中可以看到的这种要求和数据库的一个样本，但是只够在下面几章中提供不同的指导。企业广泛的发展过程将使用比较复杂的要求定义技术，如在 Oracle Designer的FHD图，有关Oracle Designer和它与Oracle Developer的使用的基本描述，请参看第15章。

第一步是评估工程的可行性。使用有限的一组要求，而且把要求的焦点放在重要的数据库上。有各种理由相信，一个简单的 Developer数据库应用程序系统完全可以满足 Talbot的要求，因为它没有什么罕见的技术要求，所以 Oracle Developer(可以建立非常大的系统的企业工具)可以容易地处理这些要求。

第二步是要得到对问题的陈述。对这个要求的简略回答是图 3-1中所说明的帐簿，它包含有Talbot帐簿所保存的在几年时间内生成的数据和业务逻辑。本书用不同技术将 Talbot的信息处理为对它当前的要求。为什么使工作内容发生了变化？因为 Talbot农场不是静止的，它们在迅速地扩展农业和工业，将农场和农业加工业扩大到全球范围。人工的 Ledger系统是为单个农场工作，但是不能为全球的农场业工作。因此，问题是使人工 Ledger系统自动化，用比当前系统大得多的能力支持企业资源计划的解决。而新系统特殊的地方又是什么呢？

Talbot需要维护和使用包含现金交易的数据库，包括货物买卖的金额、工时费的支付金额

China-pub.com
下载

以及收入的金额。Talbot还需要维护和使用包含有Talbot的工作人员和他们的地址及技能的数据库，如何转换这个功能呢？本部分将介绍一些需求的思路，比如 Talbot那样的公司的需求，以及你所需要得到的那些信息。下面的过程并不像真实的需要分析那样进行详尽无遗的讨论。

最初的推测可能是：帐簿事务是数据库的中心。在某种程度上的确如此， Talbot非常注意保持跟踪所花费和所接收的现金数量。然而，如果要更深地探究，那么就会开始认识到在事务中和人员应用中所涉及到的人的重要性。人员将系统的财务和人员的资源方面结合在一起。一个不成熟的决定是，重点表示人，将劳动者随同进入到 Talbot事务中的人员一起放入单个表中。

首先让我们列出需要强调的软件应用程序的功能。

■ 功能：表示所有的人员和机构，这个机构把 Talbot农场约束在一个主要的列表上，这个表格可用来查询和维护帐目及人员。

对于这个功能来说，最根本的是可利用这个列表来管理帐目和人员。还有一点是，与Talbot有货币业务关系的人员要在这个列表上。而且希望应用程序能够容易地修改列表中的错误。

■ 功能：跟踪Talbot涉及的货币业务，包括事务的种类(买、卖、支付、收入)、项目说明、单位的数量、每个单位的估价以及事务的数目。这些信息必须可以查询，而且必须可以被审查。

为了能够进行审查，重要的是所有事务都应存在数据库里面，而且是对独立的事务进行修正。极其重要的是，在任何时候，不论任何理由，事务都不能改变或消失。希望应用程序根据质量和价格自动计算事务的数值。用户则希望能够通过数据元素将查询的事务排序。

■ 功能：为了联系和支付工资，要保持劳动者的地址清单。

重要的是，地址应该是当前的地址，而且必须正确，希望能够为所有的劳动者产生记入标记。

■ 功能：要保持劳动者技能和收入的清单，以便分配工作。

希望所有的劳动者都有技能登记项，而且各个登记项可以评估他们在某种技能上的能力。

还有一个配置的要求：应用程序必须可以为 Talbot农场的整个网络和子公司使用。Talbot的IT部门相信，通过公司的内部和全球因特网来配置应用程序将满足广域网的需要。因此，所有开发的应用程序必须适合在因特网上使用。给出了这些要求后，就可以进行数据库设计了。

3.1.2 示例数据库

不先进行数据库设计，要使表单、报表和图形形象化是困难的。因为它们都反映了 Oracle Developer的基本目的：允许终端用户操作数据信息。数据库设计要将问题叙述清楚。格式语句对定义表单、报表和图形是很重要的。有一个示例数据库也可以使一个原型的开发工作比较容易。

附录A有一个完整的、带有数据定义语句和全部的数据的数据库。本部分内容解释数据库设计，并且讲述在以后的内容中解释 SQL及数据所需要的知识。

图3-2展示了定义Talbot数据库的统一模型语言(Unified Modeling Language, UML)图表，如果你以前没有见过 UML分类图，以下是有关的图形约定。

China-pub.com 下载

- 种类(方框)：表示数据库内的表，用表的行表示种类的对象。种类的名字在顶部，种类的属性在底部。
- 关联(直线)：表示种类之间的联系，某些关联也表示数据库的表（例如，多对多联系以及和它们自己特性的联系 ）。从地址到住所的关联上的小黑菱形表示拥有权 （正式的UML术语是composite aggregation(合成归并)），住所行拥有地址行，所以如果删除住所行，则必须删除地址行。
- 属性(方框内的文本)：表示对象的数据特性。
- 角色(名字旁边有关联线)：表示这个种类在关联中扮演的角色。
- 角色的多重性 (1和*标记)：表示在关系中和另外的实体关联的对象的数目，一个或多个。

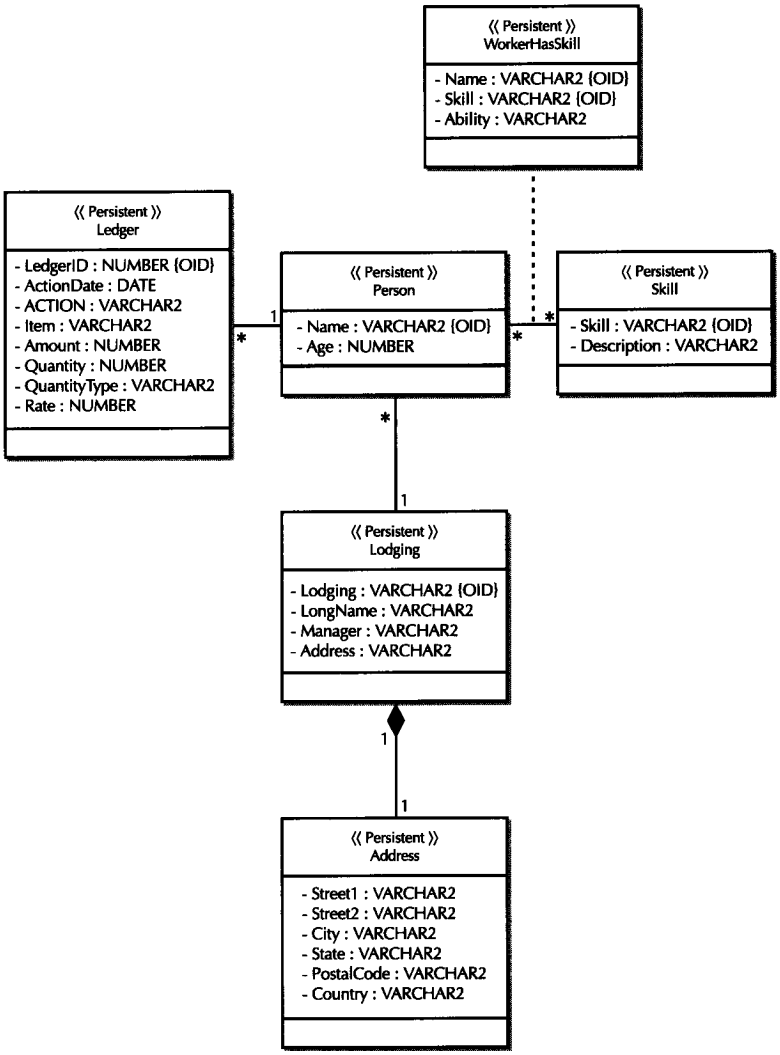


图3-2 Talbot数据库的UML种类图

有五种对象：

- Ledger：Ledger里的登记项。每个对象涉及的某种单个事务，用唯一号码标识(Ledger ID)。

China-pub.com
下载

■ Person：与Talbot打交道的人，用名字标识。
■ Lodging：某人居住的地方，由住所名标识。
■ Addres：住所的邮政信息。
■ Skill：人的技能，用技能名标识。

模型中有四个关联：

■ refers to(涉及的)：把Ledger登记项和人员联系起来， Ledger登记项涉及到人员作为卖方或收款人或付款人。
■ live at(居住在)：把人和住所联系起来，人居住在住所。
■ has address(有地址)：把住处和邮政地址联系起来。
■ worker has skill(工人技能)：把某个人和一种或多种技联系起来，或者将一种技能和一个人或多个人联系起来。能力是个单一属性，即表示工人在技能方面的能力 Ability，这是一个关系属性。

头两个关联是可以选择的一对多的关系。 Ledger登记项准确地指向某个人，而一个人可能涉及到Ledger里的多个登记项。一个人居住在某个住所内，而一个住所可能有多个人居住。在关系数据库里，表示人员的数据行包含有和住所有关的外关键字列。而表示住所的数据行则包含有与人有关系的外关键字列。在对象关系数据库中可以有外关键字，或者有连接这些数据行的参考(REF列)。有关这两种方法的完整方案信息请参看附录 A。

第三个关联需要一对一的关系。一个住所有一个确切的地址，而每一个地址则精确地对应一个住所。在关系数据库中，很可能要把地址列移到住所表中。在对象关系数据库中，可能在住所类型中创建一个地址类型和该类型的列，有关这两种方法的完整的方案信息参看附录 A。

第四个关联是可选择的多对多关系。一个人可能有几种技能，而一种技能则可能对应到多个人(多个人可能具有同一种技能)。功能只对应到工人身上——这个特殊的数据库并不表示这个事实，因为它有损于这个例子的清晰度。

将这些转换成关系数据库或对象关系数据库，要采取以下步骤：

1) 为每个实体(Ledger、Person、Lodging、Address和Skill表)创建一个表或对象类型，根据属性创建列，并且给出它们的主关键字列 (分别是LedgerID、Name、Lodging，Address和Skill)。

2) 为每一个多对多关系(Worker Has Skill)创建一个表或对象类型。创建任一列(Ability)为表的关键字，并且是参与这个关系的实体的两个关键字的主关键字 (根据Person和Skill分别是Name和Skill)。

3) 对于每一个一对多关系，在多边关系的表增加一列，这与单边关系的表的主关键字有关(在Ledger表上的Person涉及Person表，而在Person表上的Lodging涉及Lodging表)。对于关系数据库用标准的外关键字，而对象关系数据库则用参考 (REF)列。

4) 对表之间的一对一关系，变换的理解有点困难。首先，要决定这两个表中哪个是关系的拥有者。例如，在Talbot数据库中，住所拥有地址(那就是图3-2上的小的黑菱形表示的意思)。创建Lodging的关联表，要从Address中移动一列到Lodging表上，而且无单独的Address表。要创建这种结构形式的对象关系，先创建 Lodging和Address对象类型，然后，在类型为 Address的Lodging表上创建Lodging Address列。

在这个示例数据库中，要创建关系数据库或对象关系数据库，使用这些规则就足够了。

China-pub.com 下载

在比较复杂的数据库中，还有另外一些事情使人烦脑。比如，两个以上的实体的关系以及带有属性的一对多关系。使人烦脑的是还有更多的约束，而且还要考虑数据类型，有关这些问题的详细讨论请参看我写的书《Database Design for Smarties》(Morgan Kaufmann, 1999)。

如果查看附录A中的SQL，将会看到它产生一个附加的对象：序列。 Oracle提供宣布序列的能力，这是一种工具，连续地产生唯一的整数。 Talbot数据库利用这个序列在INSERT语句中为Ledger表产生Ledger标识符ledger ID。如果要求Talbot数据库可以移植到其他的数据库管理程序中，这个数据库将不包含这个序列，因为它是非标准的。不能依靠有 CREATESEQUENCE语句或NEXTVAL伪列(pseudo-column)来返回下一个序列号。 Talbot数据库有一个重要的约定，只使用Oracle。所以它可以充分利用Oracle数据库管理系统的许多非标准的特性。

注意 如果你对使用UML来设计数据库感兴趣，请参考我的书《Designing Databases for Smarties: Using UML to Design Database》(Morgan Kaufmann, 1999)或Paul Dorsey的书《Oracle 8 UML对象建模设计》。Oracle Designer工具现在对模式定义支持UML，所以还应该参考Dorsey和Koletzke著《Oracle Designer /2000 Handbook》。

3.1.3 Oracle Developer工程管理

在作任何别的事情以前，应该利用 Oracle Developer的Project Builder建立一个工程。

1. 用Project Builder产生一个工程

Project Builder是Oracle Developer工具集的一个组件，它让你把文件组织到工程里面，并且提供管理它们所需要的工具。

对于Talbot系统，我们将建立两个工程，一个对 Ledger系统，而另一个对 Worker Skills系统。要根据许多因素来决定工程的边界，包括应用程序的时间确定、你的机构如何给人们分配任务、在工程之间有多少新软件开发要共享，等等。我们将假定这两个应用程序是独立的，一个由Financial Services的信息服务(IS)小组开发，另一个在Talbot农场的人力资源内部开发。

1) 要创建一个工程最好的“赌注”是 Project Wizard。在Project Builder图标上双击，将出现如图3-3所示的对话框。

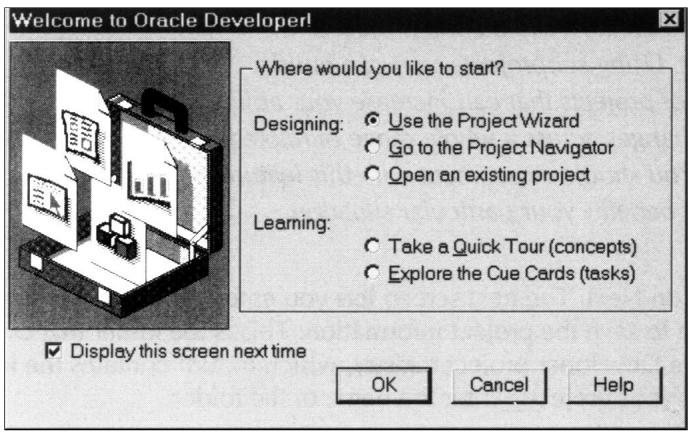


图3-3 点选Project Builder图标产生对话框

2) 在OK 按钮上单击，启动Project Wizard。在欢迎屏幕以后，出现Project Wizard对话框，

China-pub.com
下载

提示输入工程注册文件名，这个文件包含有关工程文件的所有信息和它们之间的相关性。输入文件名(见图3-4)。

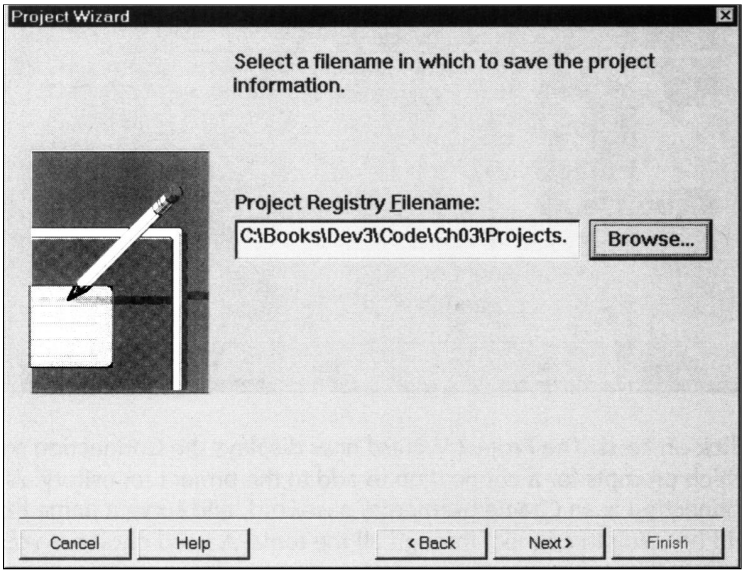


图3-4 Project Wizard对话框

注意 当有某些被定义的工程时，Proiect Wizard将插入一个屏幕，提问你是要建立一个独立应用的工程还是建立一个子工程。如果使用子工程，则可以建立工程的层次结构，可以增加管理整个关联工程范围内的变化的能力。你应该利用这个特点做试验，看其对你的特殊情况有什么好处。

3) 单击Next按钮。出现的屏幕提示输入工程的题目和保存工程信息的文件夹。该文件夹保存Oracle Developer工程注册，同样也包含所有的工程的信息。输入文件夹的名字 (见图3-5)。

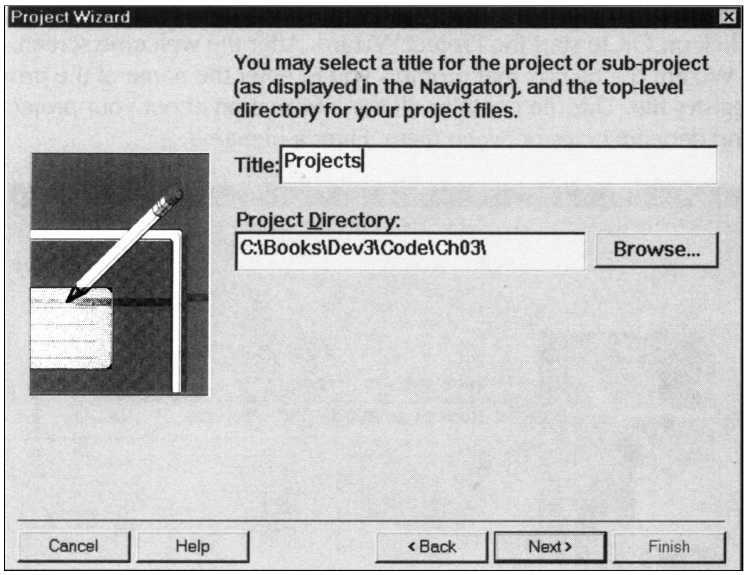


图3-5 输入文件夹的名字

China-pub.com 下载

4) 单击Next按钮，则Project Wizard显示Connection屏幕，提示在工程资料档案库中添加连接。连接内容有Oracle用户名、口令和服务名，你将通过全部工具使用连接。一个好的选择就是开发的数据库。在Talbot例子的情况下，连接就用talbot作用户名，如图3-6所示。

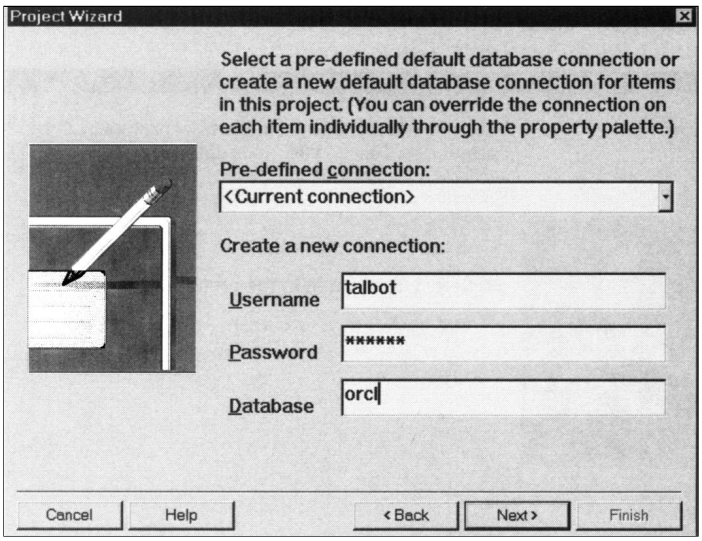


图3-6 在Project Wizard对话框中以talbot为用户名

5) 单击Next按钮，进到下一屏幕。提示输入本人的名字 (你的名字、工程管理员的名字，或者任何一个名字)和对工程的缺省连接 (如果有的话)。还可以在这里输入一段注释来说明这个工程(见图3-7)。

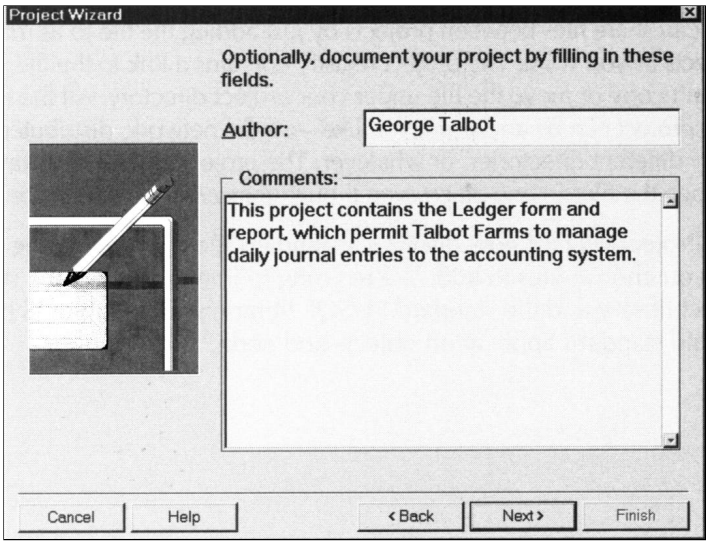


图3-7 在Project Wizard中输入注释

6) 单击Next按钮，进入最后一个Project Wizard屏幕。这个屏幕提供向工程中放置内容的三个选择：什么也不放，从Open对话框中选择的一组文件或先前在工程文件夹中选择的所有文件(见图3-8)。

China-pub.com
下载

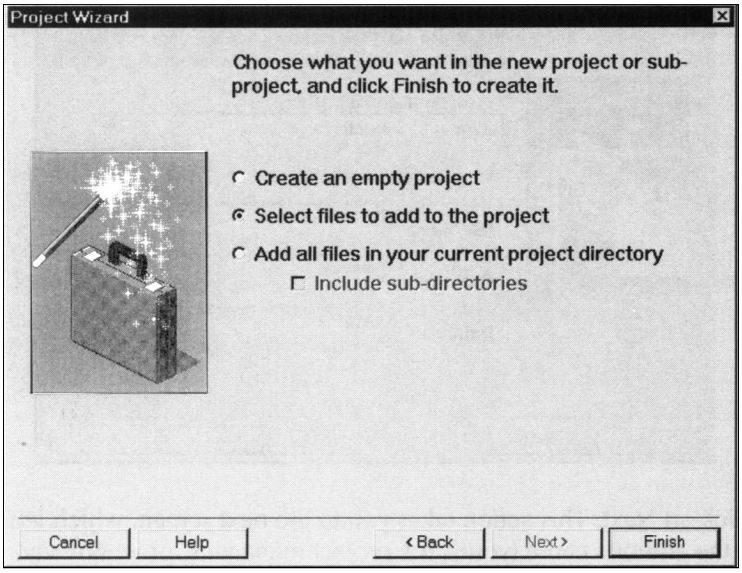


图3-8 最后一个Project Wizard屏幕

7) 单击Finish按钮，创建工程。

将文件加到你想将它们加入进去的那些工程中，就可以在工程之间共享文件。工程注册包含有对这个文件的连接，它并不在工程目录下拷贝或移动这个文件。工程中的所有文件可以出现在你喜欢的任何地方 —— 在网络上，或分散在许多不同的目录里，或无论什么地方。工程给你一种方法，即只在一个地方管理文件，即使这些文件并不在同一个地方。

Project Wizard接下来显示一个标准的打开文件对话框，提示选择要加入的文件。找到和选择标准对象库和标准的PL/SQL库，它们是Talbot IS创建用来保存标准应用程序对象和代码的(见图3-9)。

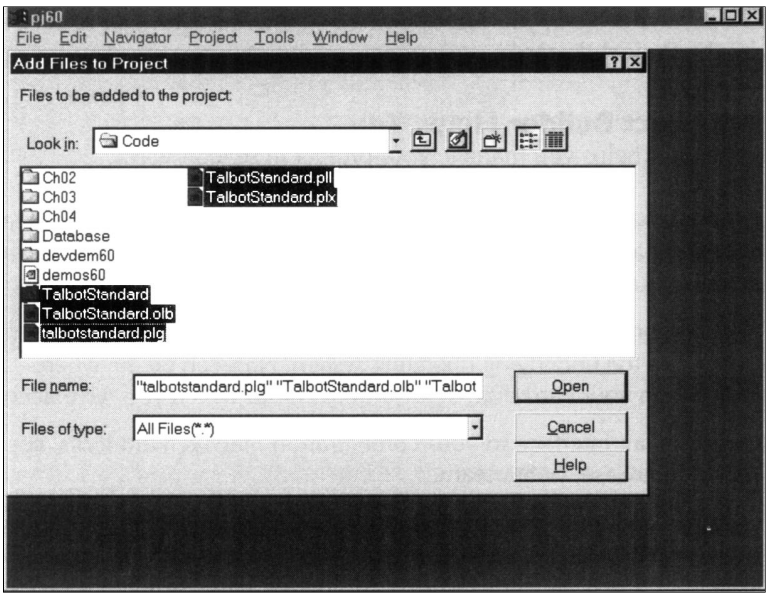


图3-9 标准的打开文件对话框

China-pub.com 下载

8) 单击Open按钮，Project Builder将被选中的文件加到工程上，这时工程已经准备好，并等待新的应用程序(见图3-10)。

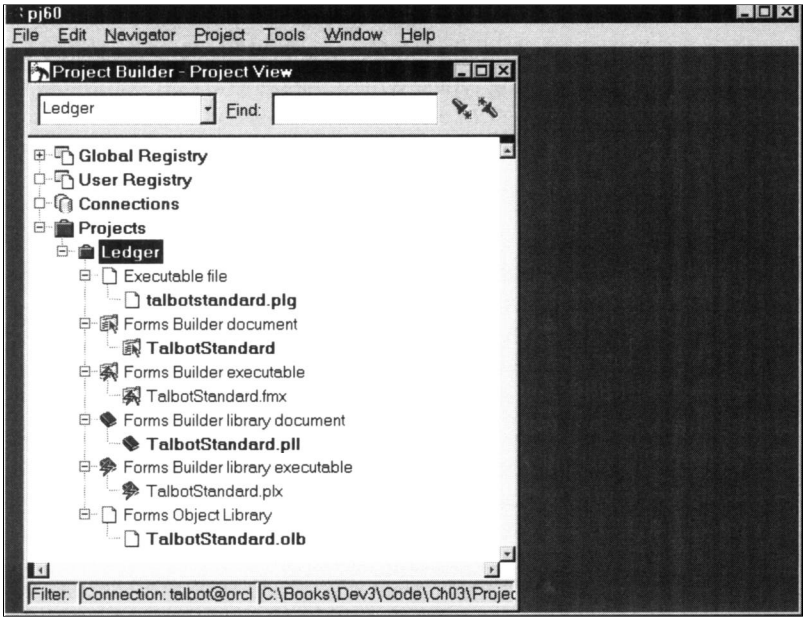


图3-10 Project Builder将选中的文件加到工程上

要添加第二个对象，只要双击Project结点，用Project Wizard启动另一个会话过程。

2. Project Builder如何帮助你

Project Builder以几种方法帮助管理工程：

- 启动各种各样的建立器。在 Project Builder窗口旁边的工具条上的建立器工具上单击，或者通过双击先前创建并加到工程中的文件。
- 可以不受基本操作系统目录结构的约束去组织文件。文件可以在任何地方——在网络上、在硬盘上，或者在所访问的任何其他地方。
- 为配置管理工具提供界面，比如PVCS、ClearCase或Starteam
- 提供一种方法，指定某个文件依赖于什么其他文件，从而自动创建多个文件。你可以改变PL/SQL程序库，例如，用一条命令可以重新建立依赖于那个程序代码的所有的表单、报表和显示。
- 通过Deliver(传递)行动使部署过程自动化，而你可以编程做适当的事情，比如把将可执行的文件压缩为一个可交付的文件。
- 因为是完全可编程的，故可以为新类型和已经存在的类型增加文件类型和各种操作。这就意味着你能够完满地定制环境以实现所希望的自动化。本书不涉及定制，有关细节请参看Project Builder的文档资料。

3.1.4 开发表单

这部分内容讲述如何根据要求着手设计各种表单和报表的入门知识。

第一个表单是Ledger表单，它着眼与Ledger有关的功能。这个表单作为会计部门与 Ledger

China-pub.com
下载

表的主要接口。这个表单必须完成以下任务：

- 表单必须允许记帐员进入会计事务。每个事务必须有日期、项目说明、活动、数量和单位(包括单位的类型、价格和总额)。
- 表单必须产生Ledger标识ledger ID以标识每一个事务。
- 事务必须通过人的名字与已经在数据库内存在的人相连接。
- 表单必须用数量乘以价格计算总额值。
- 表单必须防止对数据库中的事务进行任何更新或删除。
- 表单必须允许会计利用数据元素对事务进行特定的排序。
- 表单必须遵循适合于因特网使用的准则。

表单的域以一种可使用的格式表示 Ledger表的结构。首先要做的选择是否安排表单每次处理一个事务或多个事务。维护需求表示一个事务，而查询需求表示多个事务。单事务屏幕将域分布在表单窗口的各处，多事务屏幕则具有像电子表格式的包含字段域和数据行的网格。在本例中，选择用多事务屏幕。这种表单可以适应查询和维护。因为可以使部署图表的大小和形状适合所要求的窗口的大小。如果有更多的域，或者所需要的域比较大，那么可以建立两个表单，一个用于查询，一个用于维护。

在输入表的Person域中创建值列表(List of Values, LOV)对话框或下拉列表，可以用来满足与人员连接的要求。

第二个表单是Enter Worker Skills表单。这个表单表明第二种主要的表单是主 -从式表单。这个表单的功能是维护劳动者技能数据库，并且给管理员提供一种查询各个工人的技能的方法。引用报表也可以完成后一个任务。随后的部分给出 Ledger Summary报表的示例。

因为需求的多样性和多对多关系，最好的解决办法是用独立的表单来维护各个表 (Person、WorkerHasSkill和Skill)。Person表单包含有Person数据列， Skill表单有Skill数据列，而Worker Skill表单中有关系列(来自Person列、Skill列和Ability列的关键字)。在Worker Skill表单中，不需要包括人员的年龄。查询 Skill列和Description列的值，主关键字是叙述性的，足以作为Worker Skill表单的简短说明。可能并不需要显示 Skill Description。这样，可在表单上显示Person Name、Ability和Skill这三个域。

因为这个表单表示了一种关系，可以使用这种主 -从关系来建立登记项的自然层次结构。主记录是只读的，查询它可以找出有指定技能的人。从记录把多种技能与某个人连接起来。可以通过在这些域上的对话框List of Values或下拉列表来连接，那就意味着多次显示同一个人，以及允许多个人在同一时间得到指定的技能，那样似乎有些混乱。用层次安排则比较自然，较少混乱，实现也不复杂。对用户看起来也比较简单。因为 Person域是单独的，与技能信息的有关的两列的表不如人员和技能信息的部署图表类型的表复杂。

其他表单也许包括与Lodging表有主-从关系的Person表单，以及带有可用来输入技能的简单表的Skill表单。在表单设计过程中，可能将这些表单组合成一个更通用的人力资源表单，让Talbot的管理人员用与 Person有关的多个主-从关系去操作所有与人员有关的信息。通过使用tab canvases(标签页画布)，这个表单可以简化它的界面。 tab canvases是一种用户界面技术，可让你把几个元素放入单个视图，并通过在文件夹标签页上单击，在它们之间移动。

在创建这些表单的过程中，开始认识到要求用一些标准来确保所有这些表单给用户提供一个一致的接口。这里有一些例子：

China-pub.com 下载

- 每个表单表示一个基本的功能，使用表单的标题来概括这个功能并把这个标题放在窗口的标题中。
- 使数据块的滚动条放在这个块的右边，使它和滚动窗口的通用格式一致。
- 使用LOV(值列表)而不使用下拉列表来表示多行表中的选择项。
- 对字型、提示的放置和其他可视化特性使用用户界面标准。为了表单在因特网上的使用创造最好的效果。

3.1.5 开发报表

虽然Ledger表单对事务数据和特定查询是足够了，但是， Ledger信息的各个用户需要能够更容易理解会计汇总信息。这就需要报表而不是表单。虽然它们的区别很细微。通常，当想用一种标准的格式查询数据库的信息，但不需要操作数据时 (例如改变数据或增加新数据)则使用报表。利用Oracle Developer Server可以通过因特网使用报表，就像使用表单那样容易。所以当用户需要通过查询得到最新的信息时，应该考虑使用报表。而当他们需要增加或改变信息时，考虑使用表单。

对于那些由Talbot之外的财务人员制作的现时的会计报表， Talbot的设计者可以容易地自动产生类似的报表，而不需要在会计年审上核实这些数字而花费更多的钱。 Talbot的会计将不再需要制作报表(和承担产生报表的任务)，他们只需要做他们自己的工作。

这些报表按照事务对人的影响来汇总这个 Ledger。审查会计通过按年月顺序列出的人和对各个人的总计来查询个别事务。

付款的会计部门要按照家庭地址给工人发薪金，这就需要报表。因为 Persennel也希望能够把公司的时事通信和其他信息记入给工人的家庭地址。要做这个工作， Talbot需要根据Person和Lodging表产生记入标签。

唯一需要对记入标签报表做的主要决定是标签的大小。记入标签现在流行使用激光打印机，它包含一个相同的标记的设计图，通常每页两至三列。 Talbot Lodging Mailing Label报表需要一个设计图，说明希望 Personnel的名字和住所地址如何出现在标签页中。还可能希望能够把标签的大小当作报表的下一个参数，以及可以使用带有不同大小标签的报表。

3.2 建立表单和报表的一些规则

在商业上，表单和报表和人的关系就像图书和读者的关系那样自然。任何尝试过写书的人都会告诉你写书并不像读书那样简单。表单和报表的设计是一种艺术，它要求有丰富的经验，要了解人们对如何使用这些东西的看法。本部分内容概括了在设计 Oracle Developer的表单和报表中的一些主要原则性问题，特别是如果你打算在因特网或在内部网上使用它们的话，更要注意这些问题。

下面是五个与表单和报表设计有关的基本原则：

- 表单或报表必须用尽可能有效的方式将信息传递给用户。
- 表单或报表必须有多方面的适应性，以适应预期用户的经验水平、知识水平和不同的需要。
- 表单或报表必须尽可能地简单，而不仅仅是稍微比较简单。
- 表单或报表必须和人的响应速度和流程相适应。

China-pub.com
下载

■ 表单或报表必须通过适当层次上的反馈为它的用户提供辅助功能，当需要时，应提供帮助和提供消除错误的能力。

这些原则大多数适用于纸张表单和报表，同样也适用于联机表单和报表。适用于税款的表单(除了装有支票的税款表单以外的任何税款表单)将证明艺术在任何地方都不能接近它的顶点，以下对设计的建议可能会帮助你领会到如何使得表单更好。学习这类设计最好的方法是先看一些优秀的设计书籍，然后动手去做，而且观测用户使用这些设计的容易程度。

概括起来表单和报表设计有三要素：理解、学习和结合。理解表单或报表加速信息传递的能力，理解来自表单或报表的内在的简易性。学习如何根据表单和报表的多方面的适应性和简易性以及给予帮助的方法来使用表单和报表。如何将报表或表单安装在一起加速信息的流动以提高其性能，特别是可以让用户迅速可靠地输入信息。这些要素涉及设计的各个方面，每一个要素在用户交互系统中都起着它的作用。当然，表单和报表在某些方面是不相同的，但是它们的布局和涉及逻辑完全类似。它们的不同之处在于表单是交互式的，而报表有重复的页结构，除此之外，它们是相同的。这里讲述的设计要素，与数据登录项有关的只适用于表单，其他的要素，对报表和表单都适用。

3.2.1 通信

为了有效地通信，表单或报表只需要提供所需要的信息。要做到这一点，首先必须精确地确定有那些信息？用户将如何使用它们？有时候你会发现，将信息分成两部分，在通信上没有损失什么，却得到了简易明了的信息。这样，使用户比较容易地了解要传递的信息，从而改善了通信的性能。

作到有效通信有若干技术：尽可能少用术语。如果可能，尽量使用图形。使用标题说明或标签，避免多余信息，而且信息要一致。

1) 尽量少用术语　避免代码词，缩写词和错别字 (误用的字)。在表单或报表以及数据库的数据中使用明白易懂的英语 (或本国的语言)。如果允许向数据库输入数据，那么，可能需要提供某种编辑功能，将输入的表达为便于通信的形式。

2) 使用图形　如果图片和图形传递的信息正确，显示的层次正确，则使用图片和图形而不采用文本和表。第8章的“有效使用图形”部分比较详细地讨论有关的问题。

3) 使用标签　使用标题说明或标签能够使表单或报表的数据含义清楚。一些数据或图形可以独立，而且便于通信，但是，这种情况极少。经常可以发现标题上的几个字会立即把图形或数据的用途和性质传递给用户。

4) 尽量使冗余度最小　检查表单是否有冗余的信息。尽量减少用户输入 /输出表单或报表数据必经路径的数量。在保证整个图形的接收时，冗余技术将改善通信的性能。在吸收信息困难时，使得通信难以理解。移动数据到数据库或理解报表的内容所必须走的路径长度和复杂程度将确定通信的水平。

5) 有节制地增加多样性　有节制地利用差别来强调重要的数据或重要的指令。利用不同的字形(例如，黑体字，不同的磅值或不同的字体)可以区别标题或有关的数据，就像图形的布局那样(例如，斜体的三维区域)。但是差别太大会导致复杂性的增加，使接受信息或记忆关系的能力减小。在屏幕周围到处都有许多不同的字体、不同的颜色、有趣的符号和图标，这可能使用户感到很有兴趣(短暂地)，但是对通信并不一定有效。还有，每个平台都有风格样式的

China-pub.com 下载

要求。在不同的对象上有不同的限制范围，而且差别很大，通常会使应用程序面对互相矛盾的平台位置。

3.2.2 适应性

适应性，是指对于具有不同需要的用户，应用程序适应于他们所需要的程度。需求并不限制一个真正灵活的系统。就是说，不管计划中用户将做什么，他们可以做他们想做的任何事情。

在真实的数据库应用程序中，情况太复杂，故难以有真正的适应性。只能用编程语言或非大多数用户选择的其他类似的复杂性水平上才能得到真正的适应性。如果愿意使用 C语言或COBOL语言，那么，可以在一定的限度内做想做的任何事。但是，一个最终用户做不到，因为这些技能超出了他们的知识和能力范围。

表单或报表为最终用户提供了细心构造的、适应性较低的界面。然而，在这个细心构造的范围内，应该使用户有灵活使用应用程序的能力。

要考虑的主要的适应性问题是经验。最终用户有什么样的知识水平？当用户对应用程序的使用学得比较多时，应用程序在可用性方面可以调节吗？通常，简单的和带帮助的层次适合于经验少或知识少的用户水平，使其成为有经验的用户。应该提供一些对有经验用户有帮助的方法，比如使用关闭特性，与通过访问列表得到输入项相反而使用直接输入项、特殊按键等等。如果不能通过一个应用程序做这所有的事情，那么为不同知识水平建立多个应用程序，并且使这些应用程序相容，使得一个新用户变得比较有经验时，可以迅速地适应那些供经验用户使用的应用程序。

另外一个要考虑的主要的适应性问题是指通过可能性的网络的不同路径的潜力。人们看事物的观点不同，不应该在不需要的地方限制他们。如果输入信息的次序不重要，则应该允许可以采用任何次序输入数据。如果可以有几种方法输入一个值 (例如，以各种各样的格式输入日期)就不要限制用户只用一种格式，因为很容易检查是否正确。提供一些选项便于不同的用户用他们自己想用的方法，而不是你想他们用的方法去做事情。

3.2.3 简单性

减少复杂性就增加了简单性，反之亦然。有许多不同的原则可以用来减少复杂性。

- 可以对窗口和对话框层上的用户隐藏复杂的程度。
- 可以使得用户需要作的公用的操作简单而且容易，即使是实现这个要求比较困难。
- 可以利用人的认识能力的限度作为指导原则来限制界面中成分的数目 (在七个成分的基础上加/减两个)。
- 可以使界面成分尽可能一致，以改善用户领会这些成分的能力。
- 通过一致性原理把成分编组，使界面成分有聚合力，并且将用户需要的所有数据保存在一个地方。
- 可以使冗余度和路径的复杂性尽可能小，这也可以改善通信的性能。
- 可以使用设计布局和格式化标准，引导用户到正确的地方，使得好像是一个自然的过程。
- 通过确认和结构化输入技术(例如值列表、缺省和智能触发器)使数据输入自动化，改善

China-pub.com
下载

可靠性。

(1) 布局标准

表单和报表应该按照用户的语言和文化情况而自然布局。对于英语文化背景，就意味着把域的起点放在页面或屏幕的左上方而且从左到右，从上到下延续。

表单应该遵循平台所使用的布局标准。例如，大多数图形用户界面把表单的标题放在窗口顶部的中央，大多数平台的滚动条和状态条也有标准的位置。

为了强调，应该选定几条重要的约定，在布局中少量使用它们。记住，强调的地方越多，则被强调的地方好象是越不重要。可以使用不同的布局选项来区别表单或报表的不同元素。使标题和标签与数据不同，而且从它们概括的数据中区别汇总的数据和总计。

在文本中使用混合的字体(正常的是使用大写字母，就像英语的句子那样)比较容易读。也可以在标题或标签中使用大字字母，但是这本书建议不要这样做。在文本的标题和题目行内，避免缩写词。

(2) 结合和一致

通过把元素编组并通过在屏幕或页面上均衡分配它们，使表单和报表的布局有聚合性。

将元素分组，按照某些原则把元素结合在一起，仔细地选择原则，然后始终坚持，让布局容易被理解：

- 频率：根据用户引用的频率或进入的频率给项目分组。把最经常使用的项目放在“首要”位置，比如在英语环境的布局的左上方。
- 顺序：如果项目形成某种自然的顺序，那么按照文化背景 (例如，在英语中从左到右)的自然顺序布置它们。
- 重要性：把比较重要的项目放在一起。
- 功能：把和特定功能有关的项目放在一起。

均衡元素是一种图形设计策略，为的是让布局使用起来感到舒服。为了整齐，将域和域对齐，其他项和其他项对齐，即使是外部轮廓也一样。如果有许多不同大小的域，就有可能使用户的眼睛移来移去，这样容易分散用户的注意力。可以例外的是，在多行的文本域中使用不规则的文本是合理的。如果有这些情况的话，在大多数计算机文本布局中，右边有参差不齐的空白，其可读性比较好。而使用比例的字型的缺点可能使对齐的文本看起来很愚蠢，而且右边参差不齐的外观给用户留下更多的白色空间。最后使表单或页面的中心处于域或样板文本的重心处，这使得布局看起来比较舒服。

(3) 表单的自动化数据登录

用Oracle Developer几乎可以使表单的数据登录自动循环，但是有一些特殊的技术应该考虑。

首先，尝试重新安排表单中的数据登录域，增进其可靠性。可以使用分组 (请参看前面部分)使得登录是一个比较自然的过程。还可以利用确认来提供输入数据的立即反馈。

第二，如果可能，提供缺省值。让用户按 ENTER键接收一个值比任何其他数据的输入更可靠、更快。如果聪明地选择了缺省值，那么用户可能只需要操作不多的几项，因而节省了输入。

第三，对没有经验的用户使用值列表来提供可接收的输入。 Oracle Developer可以很容易将列表对话框和域联系起来，也容易为输入数值提供下拉列表或组合框。

China-pub.com 下载

第四，采用数据项触发器可使域变得更聪明。一种技术是依据上一个域的输入来填写另外的域，例如，采用域分组策略就能很好地工作。另外一种技术是通过智能解释输入，为有经验的用户提供输入数值的捷径。例如，可以用 PL/SQL合理地写类属代码，只用几个字符在表或记录组中查找一个值。这种“迅速—选择”方法在用户界面中越来越多地出现，而且用基本SQL的WHERE子句并不难做，可以在许多有利创造的方法中扩展这种思想，改进数据输入的速度和可靠性。

3.2.4 性能

表单或报表的性能的基本标准是操作速度和流程与操作者响应的配合程度。

表单的数据输入是关键的性能组件。如果在触发器中加入许多 PL/SQL代码，触发器在导航或确认事件过程中启动，那么要保证其性能和人的响应能够匹配。如果用户已经操作到域的外面，那么他们应该以相当迅速的次序看到确认的结果。如果这种触发器有性能问题，那么试着将这些引起问题的代码移到提交处理，或者移到在明确要求下启动而不是自动启动的触发器中。大多数用户对提交阶段的性能期望比较低，而且可以通过反馈信息，对所要求行为适当地设置期望值。把应用程序作为一个整体考虑，通常应该使用户可以预测延迟。一致的轻微延迟比有许多变量的用户不能预测的延迟要好。

在表单中查询实在不是问题，因为可以设置用户的期望值。但是如果可能的话应该避免这些情况：

- 当用户执行Execute_Query功能去查询所有的数据，而不是通过例子输入查询时，不应该长时间离开应用程序。为表单构造缺省查询，如果可能的话，它将接收一小组数据，也可以利用先于查询的触发器，如果无查询项，可加一查询项到这个查询中。

注意 许多平台提供中断查询的能力，比如在Windows或Windows 95中通过使用CTRL-C可以中断查询。可能要设置一个环境变量来做这个工作，有关细节请参看安装文档资料。

- 防止系统给用户一个不合理的或不清楚的响应。确保在大多数情况下，特别查询能力提供合理的响应。要考虑如果用户给查询加一个特殊的域将会发生什么情况。利用先于查询的触发器来预告用户对查询的响应是否会慢。

报表有三种流行形式：交互式的、特定的、批处理的。交互式报表使用按钮来提供下拉特性，制作交互的应用程序报表输出。下拉报表的导航性能和显示性能应该和响应的期望相匹配。特定的报表是当需要它们时产生的报表。这些报表的性能也应该和期望相匹配。通常要求相当迅速的响应。批处理报表可以在你晚上回家以前启动，或者通过系统中的计时器机构来关闭。对批处理报表唯一的性能要求是它们用的时间不要比转发这些报表的时间长。如果有一个日常批处理报表，那么这个报表应有比较好的格式，而且在一天之内打印。要确信你了解了所有的要求。

3.2.5 帮助

用户几乎总是喜欢在应用程序中得到帮助。有几种方法可以帮助用户。在前面部分所述的大多数布局技术，通过使表单或报表比较容易或比较自然来帮助用户工作。但是还有一些

China-pub.com
下载

更特殊的事情可以做。在交互式表单中可以做的最重要的事情之一，是立即为用户反馈他 (或她)操作的结果。这就意味着在用户的操作和信息之间，或操作和响应之间的时间应尽可能少。Oracle Developer通常提供基本反馈，提示告诉什么时间它正在工作或正在用数据库做某件事情。可以作一些工作，大大超过这个缺省特性。有两个方面要特别考虑，即确认和性能反馈。

如果用户的某个行为导致一个错误的条件，则应立即把这个错误告诉用户，有时这是比较难做的。例如，如果由于用户没有特权而改变了数据，违反了安全性，但 Oracle Developer并不能够自动地捕捉这个错误，而要等到你提交和发送数据到服务器。可以使用各种选项和技术来将反馈闭路器移动到行为上。第 9章详细讲述了这方面的内容。要确定应用程序始终在确认用户的行为的结果，或使这个结果无效。可以依靠 Oracle Developer做大部分工作，但某些情况下，可能需要做进一步的工作。

应该始终给联机用户提供足够的信息，使他 /她知道所发生的事情。Oracle Developer对几乎每一个可以想到的对象都提供广泛的帮助能力。比如项目上的工具提示。如果可能，利用它们可以为用户通俗地解释错误，而不只是给出原始错误信息。使用任何通常可以得到的帮助或互交式文档资料，为应用程序提供联机的文档资料。

最后，需要清楚地了解用户如何能消除错误。对待错误的最好技术是在第一地方 (在它们首次出现的地方)防止它们，这是技术和可靠性技术的目的。但是，极少数应用程序做到了极点，决不允许用户出错，如果是这样，这些应用程序可能不能用。应该确信理解了用户怎样能够纠正错误，然后使应用程序对用户清楚地表示出来。

Oracle Developer应用程序使用事务管理来结束数据库的改变。直到提交数据前，都可以改正任何问题。提交以后，则需要到数据库中改变数据。如果在一致性和完整性策略中有任何必然的漏洞的话，则应该为用户提供标识和纠正数据库问题的工具。

本章讲述了建立应用程序的初始步骤，而且给定了应该怎么做和不应该怎么做的基本准则，还介绍了Talbot农场的例子并用它来说明建立应用程序的第一步。第 4章将开始进入第一个教程。

China-pub.com
下载

第4章　表单、报表和图形的基础知识

现在讲述怎样开发最初的应用程序。采用第 3章创建的数据库(请参看附录A)，使用Oracle Developer Builders和Layout Editor，可以生成满足Talbot的要求的表单、报表和图形。创建表单以后，用报表Layout Editor创建报表，然后加上图形来说明表单和报表。

创建有复杂版面和许多操作的表单和报表的工作是简单的：可以方便地创建表单应用程序，显示和操作数据库中带有全部数据完整性的数据；可以非常容易地创建标准的报表；可以建立菜单并把它们连接到表单、对话框和报表上；可以容易地把图形的图表加到应用程序上。

即使用最简单的C或C++程序去做在Oracle Developer中可以做的事情，也要做许多的工作。打个比喻，就好像是做一般的饭菜 (很容易)和为美食家做美食之间的差别。用 C++编程，而且采用CASE工具编程，就好像自己饲养家禽和种植蔬菜，把它们转换成食品 (包括提供各种做法的资料)，然后在餐馆管理16排厨师。如果用程序语言编制过程序就一定能够理解，如果没有编过程序，那么试试看就会理解。

两者的差别在开发的形式上， Oracle Developer着重说明建立在过程代码上的应用程序。Oracle Developer(和本书以前的版本)的许多首次用户一开始就迫切想知道什么时候能学到他们需要知道的所有的编码技术。但这本书直到第 11章都不教PL/SQL编程，其原因是：在建立Oracle Developer应用程序的过程中，有99%的工作都不需要你去做。本章讲述如何用图形界面建立基本的表单和报表，完全不用 (或者很少)书写代码。这也是为什么说 Oracle Developer是win-win工具的主要原因：因为不用写代码，所以赢得了效率。因为得到许多成熟的工具帮助开发应用程序，所以赢得了质量。

注意　在关于数据库应用程序看起来像什么和它是如何工作的方面，Forms工具特别提供了许多假设。如果想降低生产力和指定开发小组和质量保证小组的工作，无论是从美学角度考虑还是希望增加性能，都可以使用这些不同的假设。例如要建立在单程序块中违反Forms事务句柄方法的表单；或者要建立类似于电子表格和网格的报表；或者试图利用Forms对象和Graphs对象开发复杂的图形应用程序。所有这些都可能使你处于困境。使用Oracle Developer中所预计的方法，可以在代价上和效率上获得大的效益。大多数开发人员将会发现Oracle Developer给了他们做工作所需要的工具，而且这些工具相当令人满意。

在第3章的“用Project Builder创建工程”中建立了两个工程，一个对应 Work Skill应用程序，一个对应Ledger应用程序。本章用这两个工程工作，在下面的教程中，假设正在打算使用Project Builder作为进入开发过程的主要进入点。如果不是这样，则必须用一组标准图标启动生成器，下面将告诉你在哪个工具集的工具按钮上单击这些图标。

4.1　表单教程

这部分内容来自第3章的工人技能(WorkerSkill)应用程序，这个应用程序把技能和工人联系起来。这一部分用许多插图介绍各种工具，通过实际体验，可以使你熟悉这些工具。

China-pub.com
下载

完成这个训练以后，将学到以下技能：

- 如何启动Form Builder。
- 如何用标准对象库建立开发环境。
- Form Builder如何连接到数据库。
- 如何用Data Block Wizard和Layout Wizard建立表单。
- 如何在表单中利数据库的表和列。
- 如何建立和使用表单布局。
- 如何建立和使用带滚动条的多记录块布局。
- 如何建立主-从表单。
- 如何在表单中建立数据输入项的下拉列表。
- 如何利用Layout Editor的工具调整表单布局。
- 如何从标准对象库中利用SmartClasses格式化表单对象。
- 如何将表单保存到模块文件中或数据库中。
- 如何将表单加到Project Builder Projects树中。
- 如何将表单编译成运行时的模块。
- 如何运行表单。

4.1.1 启动

首先要做的事情是启动建立表单的开发环境。

1) 单击Project Builder Launcher图标，启动Form Builder。

Builder显示Welcome对话框。选择“ Build a new form manually ”(手工建立新表单)单选按钮，打开Navigator。因为在建立第一个表单之前，需要做某些事情来初始化开发环境，通常用向导来生成表单。

注意 我不检查启动复选框上的显示，所以我见不到欢迎屏幕。对我来说，那样作只是减慢了开发过程。这纯属个人爱好。

当Form Builder窗口显示一个Object Navigator窗口时，看到一个缺省的表单 MODULE1，它将作为这个窗口的唯一表单。

现在，打开标准的Talbot对象库模块，像第10章中讲述的那样， Forms对象库模块提供一个Forms模式。可以通过子类或通过复制来用这些对象去设置属性和创建新对象的行为。大多数开发工程至少要建立一个标准对象库，使得项目、窗口、画布、警报等的外观标准化。本章假设Talbot农场用叫做Talbot Standard的对象库来做这件工作。这个对象库的SmartClasses包括为数据块项的主要类型、文档窗口，上下文画布和工具条画布、属性类和指定的可视化属性。在原形设计的处理过程中，将使用这些标准对象为原形系统创建的对象划分子类。

2) 使用OpenFile(打开文件)按钮在代码目录中找到这个对象库，并打开它。

在将来，每当启动Form Buider时，它将打开这个对象库，或打开一个最后曾经打开的对象库。为了工程的其余部分访问所有的库的 SmartClasses，必须而且只要这样做一次。

4.1.2 建立主-从表单

本教程建立的第一个表单是一个主 -从式表单。这个表单说明表单布局的两个基本类型

China-pub.com 下载

—— 表单式和表格式。表单式布局方式用单个记录布置屏幕。而表格式布局按照带有多个记录的网络格式或电子表格来布置屏幕。主 -从式表单通常以显示单个记录为主，表格式布局中是详细的一组从记录，在本教程中，表单显示某个工人，然后列出这个工人具有的所有技能作为这个Worker主记录的从记录。

1) 在Object Navigator中选择表单对象。

2) 单击对象名字Module1。

3) 键入一个新名字：WorkerSkill。

注意，Form Builder把键入的每个字母转换成大写字母。现在已经生成了WorkerSkill表单模块(见图4-1)。但它不作任何事情，只是让你喘口气而已。

图4-1 WorkerSkill表单模块

要用人工创建表单而不要进入 Data Block Wizard，因为需要建立表单的CoordinateSystem属性，像第5章推荐的那样使用英寸而不是点。这样作使得能够利用TalbotStandard对象库，因为它使用类似的Coordinate System。如果你只用过向导的话，这是展示某些你没有见过的东西的机会。

4) 双击表单图标或右击它，从弹出式菜单中选择 Property Palette，选择表单对象并显示属性设置板。

5) 向下滚动到Physical属性组，然后在Coordinate System属性上或在它的More按钮上双击，观看Coordinate Info对话框。第5章将作详细的讲述。

6) 把Coordinate System设为Real并把Real Unit设为英寸。

提示 TalbotStandard对象库的Coordinate System基于英寸，所以必须把使用这个库的对象的每一个应用程序的Coordinate System属性都改为英寸。例如，在从这个对象库划分子类的对象中，Width和Height应该用英寸表示。但是如果Forms Coordinate System是Point(点),它将把数字解释为点，而不是英寸，这种设置可能用处不大。还应该注意，与Oracle Developer一起提供的标准对象库采用的是基于点的Coordinate System，它可能适应也有可能不适应你的特定情况。如果不考虑操作系统或用户接口的可移置性，可能发现接受缺省的Coordinate System比较方便。如果这样，将不必每次经过人工创建表单，但是损失了某些可移置性。

现在已经建立了标准环境和第一个表单的基本元素，本教程的下一步是创建表单的细节。创建这个表单要经过三个步骤：

- 创建Worker数据块，从人员表中显示单个人员的记录。
- 创建Worker Has Skill数据块，从WorkerHasSkill表中显示多个技能记录，并让它把从数据块连接到主Worker块上。
- 增加一个下拉列表，用它提供 Skill表中所有的技能，使得无须键入技能而只需用从列表中选择它。
- 清理表单外观。
- 保存并运行这个应用程序，确认将它作为原形。

在开始工作之前，连接数据库。选择 File|Connect并输入用户名、口令和数据库的服务器别名(例如在这里用talbot, george和Orcl)。

注意 在Oracle Developer的开发过程中有几个访问数据库的点，在到达其中的某个点

China-pub.com
下载

之前，不需要连接。在任何要求连接数据库的点上，如果没有连接，则 Oracle Developer自动使用Connect对话框提示连接。

1. 创建Worker数据块

Worker数据块对应于数据库中的 Person表。通过查询这个块的记录，可以滚动通过这个数据库的所有Person的记录。WorkerSkill用来给工人赋予特定的技能。必须要做的第一件事是在数据库中找出要赋予技能的那个人。在这种情况下所需要的是 Worker名字的单个域。

首先，启动Data Block Wizard，当第一次进入Form Builder时，如果选择了Wizard选项，那么这个过程会自动产生。但是因为是用人工建立表单，所以必须自己启动。

1) 下拉Tools菜单，并选择Data Block Wizard项，Builder显示Data Block Wizard的欢迎屏幕。如果在使用向导建立表单的屏幕上选择了这个屏幕，就总是从 Form Builder Welcome屏幕直接进入这个屏幕。

2) 在Next上单击，所看到的屏幕让你在建立在表的数据块或在存贮的程序之间进行选择。应选择Table或View选项，如图4-2所示。

提示 有关使用存贮过程建立数据块的细节，请参看第6章“过程的封装”小节。使用存贮的过程在数据的访问上有更多的控制，并且通过减少网络的信息量而提高了性能。可以在使用Oracle作为数据库服务器时使用这种方法。然而，也可以通过建立一个新的数据块来启动Data Block Wizard。在Object Navigator中选择Data Block标记并在Create工具上单击，Oracle将提示是使用Wizard还是人工来建立数据块。

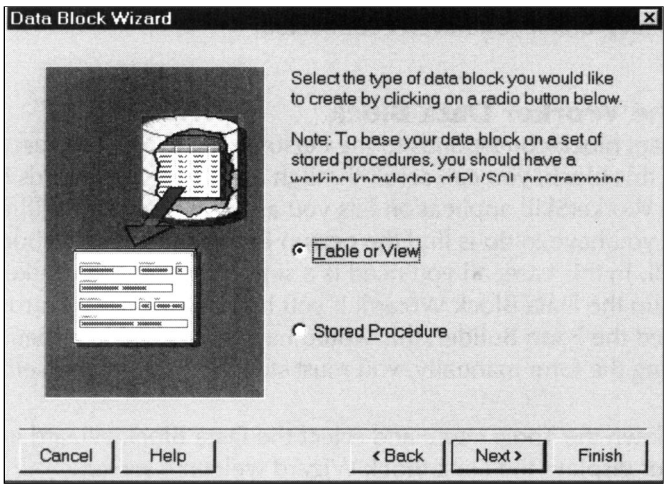


图4-2 Data Block Wizard的典型屏幕

Data Block Wizard现在显示表屏幕，它要求输入表或视图的名字，并选择表的列或视图的列，以便将它们加入到数据库中。

3) Browse按钮让你浏览表的数据目录。在它上面单击， Builder显示Tables对话框，问你想看什么(Tables、Views或Synonyms)和来自什么模式(CurrentUser或Other users)，然后它显示一个对象列表，可以从中选择当前用户的表 (见图4-3)。

4) 选择想要的表名字“ Person ”，输入到Base Table域中。只要在相应的那一行上单击，然后单击OK。

China-pub.com 下载

在图4-4中的Wizord的数据块屏幕上，Data Block Wizard假设数据块和所选择的表使用同样的名字，所以在这里不能改变它。极力推荐你保持名字相同，使得以后容易了解正在进行什么。数据块屏幕列出表的各个列，这些列包括Name列和Age列，外加两个独立的住所列。Person表的Lodging列实际上是Oracle 8的REF列，它指向住所表的一个数据行。Oracle Developer知道这种关系。列表中的第一个Lodging列是Lodging表的查找列，可以单击符号“十”部署列表，而且可以在数据块中包括这些列作为数据块的项。如果这样做了，注意到将又有另一个REF，这一次是指到Address表，而且可以把任何Address列插入数据库块中。于是Oracle Developer通过Data Block Wizard赋予你访问Oracle 8对象特性的全部能力。

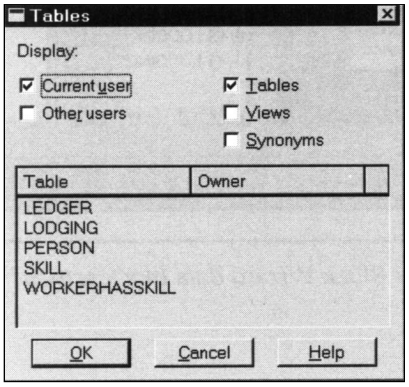


图4-3 Tables对话框

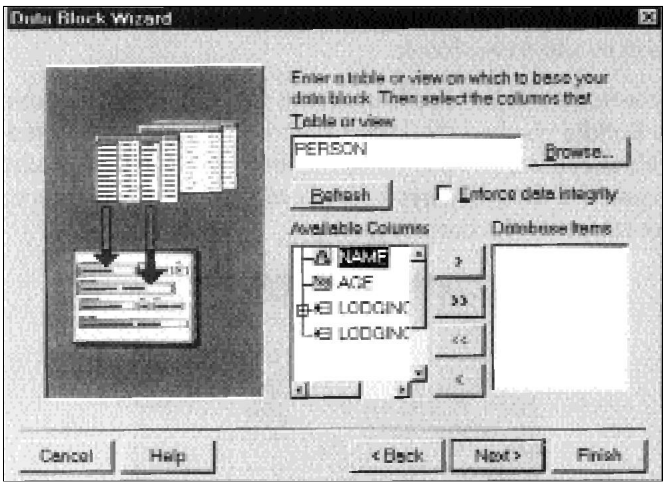


图4-4 Data Block Wizard数据块屏幕

提示 数据库可能已经变化而你不知道。如果在Form Builder中做完工作返回到这个屏幕，而且不是第一次进入它的话，可以使用Refresh(刷新)按钮，确保所看到的是所有有效的数据列。

5) 使用>和>>按钮，可以将数据列移动到Database Item列表中，这个列表是你想在数据块中创建的项目列表。如果只要Name列，就在那列上单击选择它。然后在＞按钮上单击把它移到Database Item列表中。或者在Name列上双击，再单击Finish(完成)。

6) Oracle Developer现在启动Layout Wizard，显示欢迎屏幕。在Next上单击，出现显示图4-5的画布屏幕。这个屏幕缺省地为这个数据块建立一个新的内容画布，这就是你想要的。

7) 在Next上单击，进入到数据块屏幕。在Available项目列表中看到NAME项，双击它告诉Wizard表示要显示这一列。于是激活了Item Type域，缺省时是Text Itme，如图4-6所示。在Finish上单击。

Worker数据块接着出现在WorkerSkill表单下的Object Navigator上，Layout Editor显示这

China-pub.com
下载

个新布局，祝贺你有了第一个数据块！

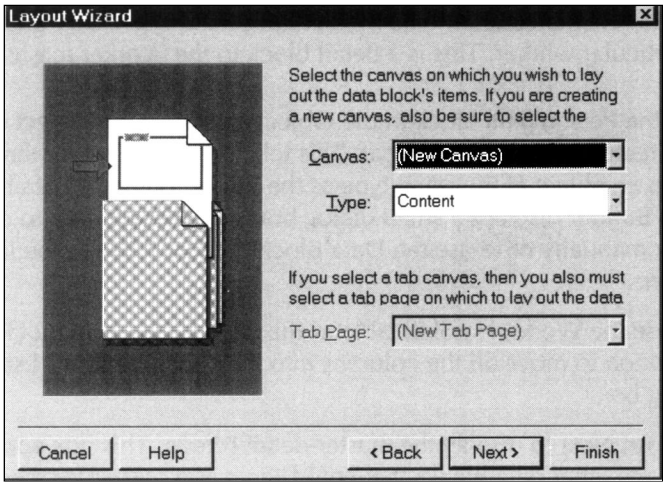


图4-5 Layout Wizard画布屏幕

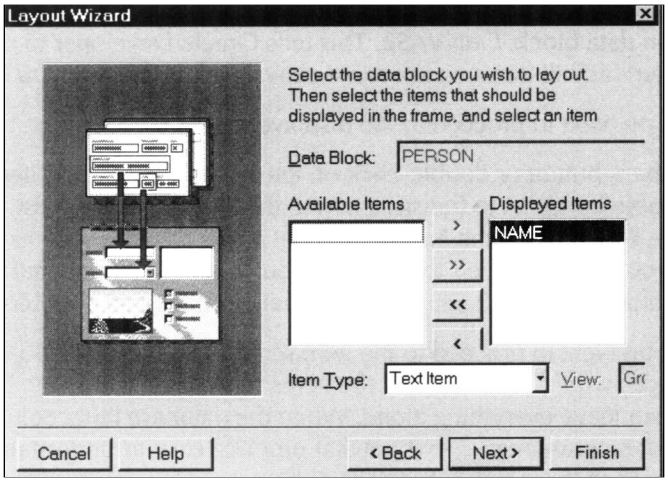


图4-6 Layout Wizard数据块屏幕

2. 创建WorkerHasSkill数据块

现在，创建WorkerHasSkill数据块，用来让用户查询和输入工人的技能，这是 Worker主数据块的从数据块。

1) 在Object Navigator上找到Person数据块并选择它。然后单击工具条上的 Create按钮，告诉Form Builder你想创建类似于Person的类型的对象——数据块，所以Form Builder显示一个小的对话框，问你是用人工创建数据块还是使用 Data Block Wizard。选择后面一个单选按钮。

2) 选择WorkerHasSkill表作为这个数据块的基础，利用 >>按钮把所有的列移到这个表对话框的Database Item列表中。

3) 在Next上单击，显示主-从屏幕。如果有另外的数据块可以作为你要创建的下一个主数据块时，这个对话框才出现。当你产生表单的第一个数据块时，你看不到它。

China-pub.com 下载

4) 在Create Relationship按钮上单击，显示可以得到的主数据块的列表——Person列表。

5) 在列表的OK上单击，Builder用缺省的连接填入其他的域，通过人员名字把人员和技能连接起来，如图4-7所示。单击Finish完成数据块并启动Layout Wizard。

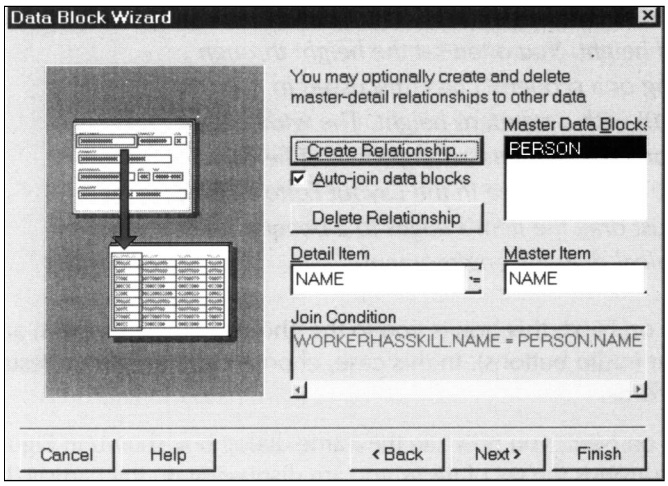


图4-7 Data Block Wizard主-从屏幕

6) Layout Wizard的画布屏幕显示出为Person数据块创建的画布CANVAS2，告诉Oracle Developer在同一个画布上创建WorkerHasSkill项。

7) 在Next上单击，前进到项目显示屏幕。

8) 使用＞按钮，或者双击Available Items列表中的SKILL和ABILITY项，把它们传递到Displayed Items列表中，如图4-8所示。使NAME项不进入Displayed Items列表意味着在画布上看不到它。你已经在Person数据块中看到了名字，所以在从数据块中不需要看见它。

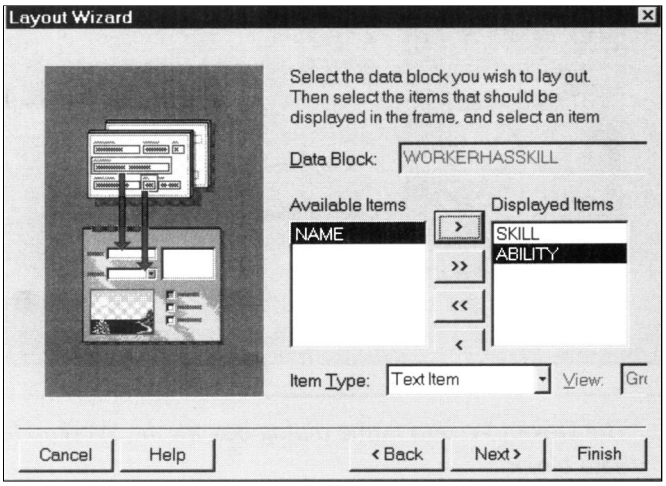


图4-8 WorkerHasSkill数据块的Layout Wizard项目屏幕

9) 在Next上单击，前进到项目屏幕中。在这个屏幕中修改所选择显示的项目的提示、宽度和高度。可以让每一个项目独立安放。当数据库表列的名字不是自然语言的单词时，制作一些容易理解的提示作为这儿的提示。

China-pub.com
下载

提示　我发现等到以后再修改宽度和高度比较好。你可能经常用划分子类或属性类(在第10章中讨论)的标准高度来设置高度。宽度取决于最后的域的安排，这可以在 Layout Editor中看到。在图形绘制环境，只能拖动项目的边缘来设置较好的宽度。

10) 在Next上单击，然后在 Form和Tabular布局的单选按钮间进行选择。在这种情况下，选择Tabular，即选择用网格表示记录。

11) 在Next上单击，出现如图 4-9所示的框架对话框。框架把画布上显示的一组项目进行分组。以后，可以通过选择这个框架，用 Layout Wizard来修改这些项目。

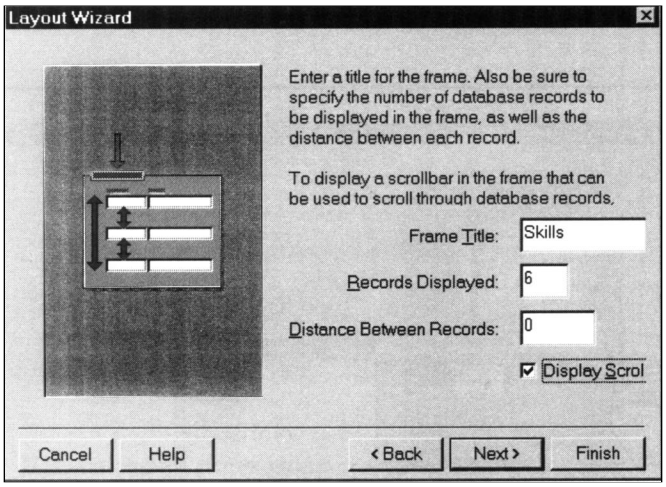


图4-9　Worker Has Skill数据块的Layout Wizard框架对话框

12) 输入框架的标题“ Skill ”，并设置Records Displayed为6。核对Display Scrollbar复选框，显示在标签页上的六个记录，如果有六个以上的记录的话，用户可以用滚动条滚动观看。在Finish上单击。于是，你没有编写任何程序语句代码，就用 Layout Editor得到了一个主-从表单。

注意　设置Distance Between Record是要在画布上每两个记录之间用标准的数字分开。这个特性的主要用途是允许记录换行(Wrapped)。这些记录卷绕显示若干行上的数据项而不只是一行。通过留出行间的空间，可以移动项目组使它们重叠，把记录第二行的项目放在记录第一行项目的正下方，这可以使你在相对小的水平画布上显示长记录。用一些设置来进行试验并在Property Palette上查看这个相对专业化的特性的作用。查看每一项的提示属性，把项目的标签说明放在满意的位置上。标签是项目的属性，不是画布上独立的样本正文。

3. Skill列表

为了完成第一个表单的工作，现在将一个列表放入 WorkerHasSkill数据块，以便帮助用户输入技能数据。这项工作有几种不同的作法，但最有效的方法是让 WorkerHasSkill数据块的Skill项成为列表项。

列表项将列表数值呈现给用户，让用户从中进行选择。在这个列表可用的情况下，用户并不需要记住要键入的准确的数值，因此它避免了键入错误和随后的改正，所以它改善了性能。可以用两种方法建立这样一份列表：通过直接把值输入到表单，或安排表单从数据库中

China-pub.com 下载

查询这些值。后一种方法要灵活得多，因为可以增加新值、改变现有的值或从数据库中删除这些值，而不用重新编译表单。可以建立这种动态列表作为 GUI的标准的下拉列表或 Oracle Developer值列表对象。在这里，因为这个列表相对比较短，所以选择下拉列表的格式。

提示 有关使用LOV Wizard建立广泛使用的列表建立LOV的例子，请参看本章后面的“利用LOV向导”部分。LOV的方式有很好的模式匹配搜索能力，这使得它适合作长列表选择。用户只要输入一两个选择字母，LOV就能发现它，因此即使LOV不是标准的GUI控制，但是长列表选择的工具。

这种方法要求使用少量的 Talbot农场的PL/SQL程序库的标准代码，所以你必须首先把这个程序库连到这个表单上。

1) 在Object Navigator中找到这个要连接的Attached Libraries的结点，并在其上双击，把这个库与表单连接。

2) 把这个库的名字“Talbot standard”输入到Library域中(见图4-10)。

3) 在Attach按钮上单击，Oracle Developer在System Registry中搜索标准路径FORMS60_PATH(请参看这一段后面的注释)，然后将在Attached Libraries结点下出现TALBOTSTANDARD库和它的子程序。Talbot农场有一个简单的程序叫做Set_Up_List处理建立database-driven列表。

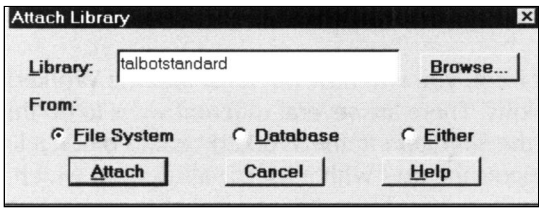


图4-10 在Attach Library中输入库名

注意 在连接这个程序库之前，需要将这个库的路径加到记录上。找出 TALBOTSTANDARD.PLL文件并记下到它的文件夹路径，在 HKEY_LOCAL_MACHINE|SOFTWARE|ORACLE记录文件夹下编辑记录关键字FORMS60_PATH包含那个路径。如果不熟悉这些环境变量的信息，请检查你的操作系统的文档资料。也可以把全部路径放入库的域内，但是这是一个不好的做法。如果以后把这个文件移到另一个目录，那么必须改变这个路径并重新编译表单。如果不规定路径，就不用做这额外的工作。

现在需要对项目划分子类。

4) 在WorkerHasSkill数据块项目组下，找到Skill项。

5) 在这个项的图标下双击打开Property Palette。

6) 在Subclass Information属性上单击两次使Skill成为一个子类。

7) 在Subclass Information对话框上，从下拉列表中将Module名设置为TALBOTSTANDARD。然后把Object Name设置为LIST并单击OK，使这个列表成为Talbotstandard程序库标准列表对象的子类(见图4-11)。

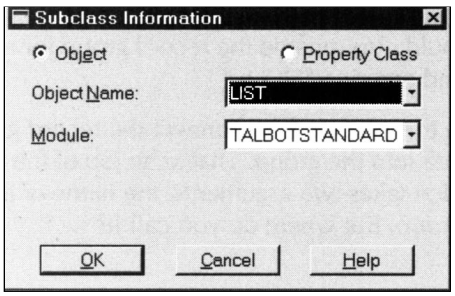


图4-11 Subclass Information对话框

建立Skill列表后的下一步是创建记录组，记录组是类似于表那样的对象，它包含项目的多个记录。用记录组来作为在这个列表中显示的项目的存贮区。当表单启动时，你希望Oracle Developer通过从数据库的Skill表中对Skill的查询建立这个记录组。当用户

China-pub.com
下载

下拉Skill列表时，列表项必须访问记录组的这些记录，以产生在下拉列表中显示的项目列表。

8) 找到Object Navigator的Record Group标题，并双击它，产生新的记录组。当这么做时，Oracle Developer提示使用查询。

9) 输入SQL SELECT语句，查询Skill表的所有技能：

```
SELECT Skill, Skill
  FROM Skill
 ORDER BY 1
```

这个SQL语句必须用专门的格式，因为要用它填写列表。列表对每个列表元素有两个值：一个标签和一个值。标签在下拉列表中可以看到，值由 Oracle Developer放入数据库。这样，显示的是可读的文本字串，放入数据库的是数码或某些其他的值而不是全部的文本串。有关细节请参看第6章的“列表”部分的内容。

这个SQL语句因此不能只检索单组值，它要按照排序检索标签及其对应值作为 SELECT列表的两列。因为Skill表用文本串表示技能，所以标记和值可以选择用同一列。这样， SELECT列表两次选择Skill列——第一次是给标签，而第二次给值。 SELECT还根据第一列的标签将结果排序，所以标签列表将按字母数字的次序显示。当单击 OK时，Oracle Developer分析这条SQL语句，确保正确之后，产生记录组。为了可读性还应该通过在名字上单击并输入“ Skill”重新命名这个记录组。

这个过程的最后一步是将列表项连接到这个记录组，并输入数据到记录组中。这个工作由Set_Up_List程序来作，就是要得到两个参数：列表的名字和记录组的名字。那么，到哪里去调用它呢？

通常是在启动表单时建立列表项。因此将启动时运行的 PL/SQL程序的调用放在表单的触发器上。

10) 在表单的下方选择Triggers标题。

11) 右击这个标题图标弹出它的菜单，并选择 SmartTriggers，你会看到几种可能性。

12) 选择When-New-Form-Instance触发器项。Oracle Developer打开PL/SQL编辑器，等待编程。

13) 为输入一个对Set_Up_List的调用，找出Object Navigator的Attached Libraries结点。扩展TALBOTSTANDARD程序库。

14) 在工具条上用Expand ALL工具为Set_Up_List扩展所有的结点。

15) 在Specification Heading下选择全部的说明。

16) 从主菜单中选择Navigator | Paste Arguments菜单项。

17) 回到PL/SQL编辑器，会看到所需要的程序调用 (见图4-12)。

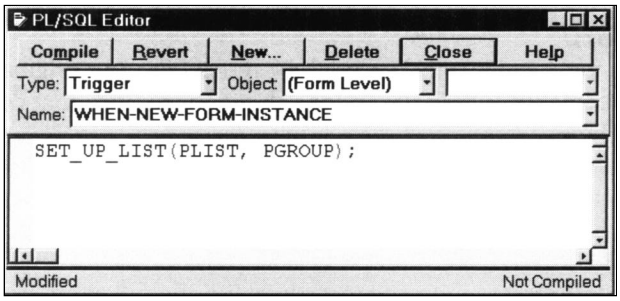


图4-12 PL/SQL编辑器

China-pub.com 下载

18) 用列表项的名字‘ WorkerHasSkill ’替换字符串PLIST，用记录组的名字‘ Skill ’替换字符串PGROUP。不要忘了字符串上的单引号“ ’”。这个单引号指出这些是字符串，不是变量，而且与字型(大写、小写或混合型)没有关系。

19) 单击Compile，确认所做的每一件事都是正确的，编译这个代码。

从现在起，当运行这个表单时，触发器将启动并自动地填充 Skill下拉列表。

注意 用它的块名限定项名是个好的想法。因为这时表单上只有一个Skill项，所以在这里并不严格需要这样。如果以后你用 Skill项增加另外一个块，那么代码可能不能像它以前那样工作。

下面需要建立各种列表项的属性，那样记录组就可以填充这个列表了。

20) 在WorkerHasSkill数据块的Skill项的Elements in List属性上单击，将看到从对象库继承的单个列表元素“ Default Value ”。

21) 改变List Item Value域成为数据库的表的一个有效值。即输入“ Work ”并单击OK。

22) 在Mapping of Other Values属性中输入同样的值：“ Work ”。

4. 表单是“ 天生的 ”

你一开始就有一个表单，不过它看起来可能完全不像你最初的设想。 Oracle Developer对这个表单的外貌作了一些假设，这些假设不需要与你的要求匹配。

为了查看这个表单，双击 Canvas图标打开画布(Forms | WORKERSKILL | Canvases)。Builder显示Layout Editor，把画布的样子展示给你，如图4-13中所示。

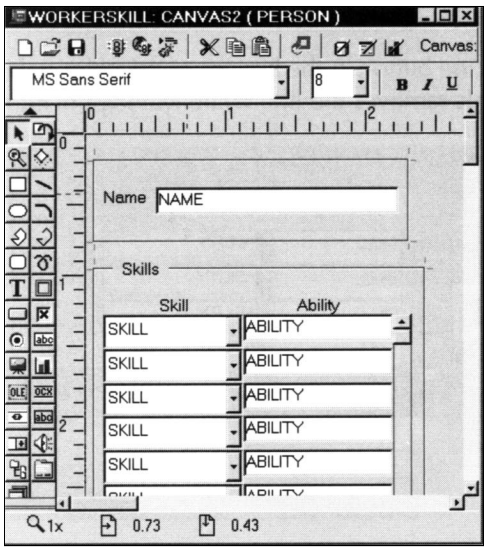


图4-13 WorkerSkill画布的缺省Layout Editor

为了完成表单设计，需要在画布上完成以下格式变化：

- 除去框架线和标题。
- 从标准对象库划分子类。
- 把Name列标签名改为“ Worker ”。

1) 要除去线框和标题，在 Layout Editor上单击，选择frame(框架)。

2) 使框架紧紧包围这些对象，然后显示框架对象的Property Palette。为此可以在被选择的框架上双击；或者可以在选择的框架上右击，弹出菜单并在Property Palette上单击，或者从主菜单中选择Tools | Property Palette。

3) 找出Layout Frame Group,设置以下的属性如图4-14所示：

Horizontal Object Offset	0
Vertical Object Offset	0
Allow Expansion	Yes
Shrinkwrap	Yes

图4-14 设置Layout Frame Group的属性

- Horizonal Margin (水平边距)：0
- Vertical Margin (垂直边距)：0
- Horizonal Object Offset (水平对象位移)：0
- Vertical Object Offset (垂直对象位移)：0

China-pub.com
下载

■ Shrinkwrap (卷回)：Yes

4) 重新定位框架使它的角在画布的右上方，这种定位使Name域在画布上占据的空间尽可能少。

5) 在工具调色板的底部找到Line Color工具并单击。

6) 在弹出的颜色调色板的底部找出No Line按钮，并在其上单击，就除去了线，而框架还在原来的地方(见图4-15)。

所看到的框架像一条蓝色的阴影线，在运行应用程序时将不再出现，如图4-16所示。

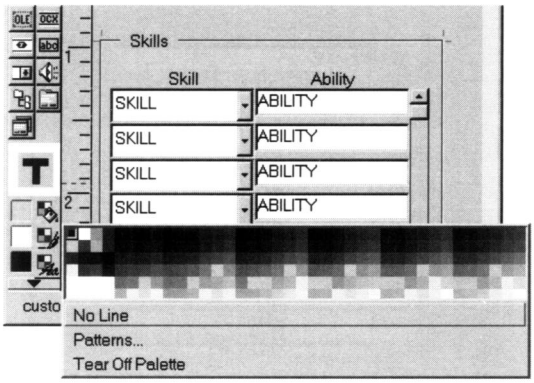


图4-15 颜色调色板

图4-16 框架像一条蓝色的阴影线

警告 虽然按DELETE键可以删除框架，但是决不要这样做！为了以后在框架元素上使用Layout Wizard，需要保留框架。如果删除它，就不能再利用向导系统来改变数据块或它的布局，这样就丢掉了Oracle Developer最好的一个特性。

对Skill框架重复这个步骤，把这个框架直接定位在 Person框架下。还需要除去Skills框架的标题。

7) 显示这个框架的Property Palette。

8) 找出Frame Title标题并在Frame Title属性中除去“ Skills ”字符串。返回去查看Layout Editor，将发现框架看不见而且没有标题了。

注意 除去框架线和标题是个人的偏爱问题。在这里，基本布局如此简单，这些行只占据空间，没有任何可见的值。很容易区分skill表格和单个name项。对于比较复杂的页面和画布，用可见的框架线条包围数据块的项，将它们和同一画布上的其他数据块区分开就有意义了。在设计项时，直观地区别屏幕上相关的信息簇，可以给用户一个组织数据的直观线索。在各种情况下，要熟悉这些属性，因为需要了解在不同的环境下，它们是如何影响画布的。

要使用Talbot标准格式正确地格式化项目，就要对用SmartClasses创建的所有项目划分子类。 SmartClass是在打开的对象库中的一个对象，这个库的创建者把它标记为标准对象(有关细节请参看第10章)。当在Object Navigator中选择对象时， Oracle Developer在打开的对象库中创建一个标记为SmartClasses的对象的列表，它和你选择的对象是同一个类型。对这个对象划分子类所需要作的全部事情是右击鼠标并从SmartClasses集中选择弹出菜单。

9) 在Person Block选择Name项。

10) 在这个项上右击并在SmartClasses菜单项上单击，将看到几个类名。

11) 选择Text，使对象是字符型文本项(见图4-17)。

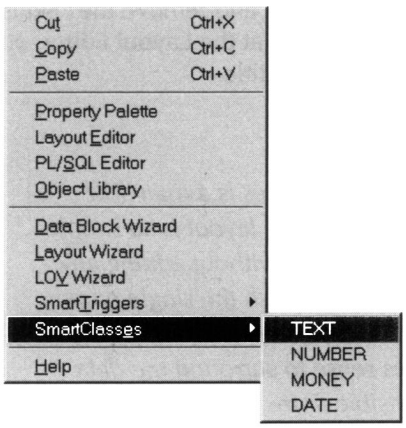


图4-17 选择Text选项

China-pub.com 下载

12) 对WorkerHasSkill数据块的Name和Ability项，重复这个步骤。对WorkerHasSkill数据块的Name项并不真的需要再分类，但是不管怎么样，在它继承或将来要继承一些特殊的属性的情况下，对它划分子类是个好主意。

注意 在SmartClass菜单中看到的东西完全取决于建立和打开的对象库。在这个例子中，画面显示Talbot Standard对象库中被标记为SmartClasses的三个对象，如果使用的对象库，可能会看到某些不同的对象。

当划分子类项目时，库对象的各种属性值取代子类对象的对应值。例如，如果文本项有一个标准的宽度和高度，那么 Oracle Developer把画布上的这个项的尺寸重新定为标准宽度和高度，而忽略 Layout Wizard的设置。然后，可以通过 Property Palette或通过Layout Wizard再废除子类的设置。也可能要垂直地或水平地拖拉框架以适应这个重新规定尺寸的对象。

当要改变从另一个对象继承来的属性时， Property Palette显示一个小图标，告诉你取代从父对象继承的属性值的值。始终可以用在 Property Palette顶部的继承工具恢复这个继承值。选择要继承的属性，并在这个工具上单击，则这个值变回到父对象的值。

13) 为了改变Name的提示，再次找出Name项，并显示Property Palette。

14) 找到Prompt的标题并把Prompt属性的字符串改为“ Worker ”，即改变了Layout Editor中的提示。

注意 可能需要稍微改一下框架的尺寸以适应新的提示。关闭框架的 Shrinkwrap属性，然后在Layout Editor中选择框架，并把框架拉出文本框的右边框。在移动这些组件之前，用View |Snap to Grid菜单项显示Layout Editor的snap-to-grid特性。Layout Editor假设网格放在画布的下方，并且在画布左上方显示标尺。可以用View | Grid菜单项显示网格，背景填充会使它变暗。使Snapping为on，当移动各种组件时，通过查看标尺，可以容易地将它们按照象素排列。用Ruler Settings对话框(View | Settings | Ruler菜单项)把网格的单位改为英寸，使得可以和你的Coordinate System一致。我发现每个网格空间为0.25或者为4个snap点最适合我的应用程序样式，在完成了周围的设置以后将框架的 Shrinkwrap属性恢复成为yes。

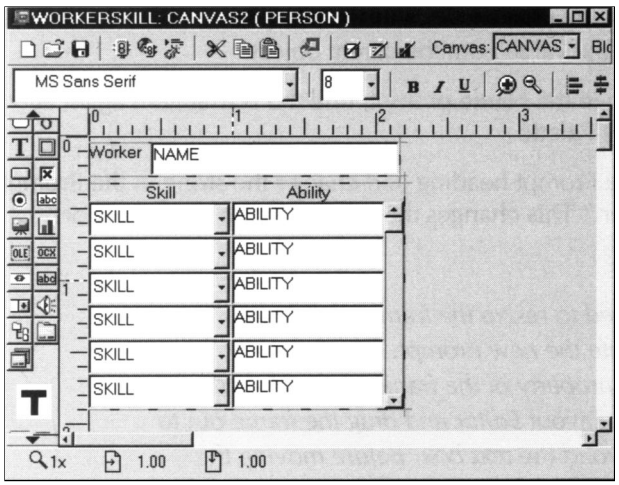


图4-18 在Layout Editor中修改的WorkerSkill画布

这个过程结束时，画布看起来应该像图4-18显示的那样。当然完全取决于你是如何进行修改的。

4.1.3 保存表单

现在有了一个完整的、可以完成第3章安排的任务的工作表单。可以利用这个表单来查询，

China-pub.com
下载

包括通过例子来查询数据子集的查询。可以使用这个表单来操作数据：增加新记录、查询记录、删除记录或在输入数值去改变已经在数据库内的数据，而且仅需用几分钟的时间就可以做所有这些工作！

现在应该保存所作的工作，运行应用程序，查看表单的实际模样，确定它是否如你想象的那样工作。

可以把表单模块保存在模块文件中或保存在数据库服务器上的应用程序的中心数据库中。我建议你用前一种方法。当进行这个决定时，要考虑的问题主要有：

- 模块的可接近性：如果你有一个相对复杂和大的应用程序环境，并希望和其他开发人员共享你的模块，那么把这些模块保存在 Oracle Developer可以访问的数据库中要比把它们存在文件中要好。在网络上共享文件是可能的，但是可能比较复杂，特别是在广域网中。
- 工程管理：利用Project Builder可以管理文件，而不能够管理数据库中的模块。这就意味着如果把模块存贮在数据库中的话，那么将失去 Project Builder的工程管理和部署特性。
- 完整性：与任何数据一样，有一个数据完整性的基本问题，就是说，如果发生媒体损坏或别的某种数据毁损事件，那么可能会丢失应用程序模块。因为数据库提供很强的完整性机制，如果你把模块存贮在数据库中，则风险较少。备份文件或定期备份数据库，可以恢复丢失的数据。
- 性能：存入数据库比存入文件慢，而且恢复也慢。
- 配置管理：如果想使用一个商业配置管理系统来管理你的应用程序，则必须使用文件，因为这些系统还不能用数据库工作。
- 方便：可以使用标准的文件系统实用程序来操作文件。数据库表和工具的灵活性比较小，在数据库中模块的名字可以有 30个字符。
- 可移置性：可以把模块只保存在 Oracle数据库服务器中。如果打算在另外一个数据库上开发，那么必须使用文件来存贮这些模块。

因为这本书极力建议使用 Project Builder和配置管理系统作为风险管理过程的一部分，而且因为各种方便和性能问题，建议使用文件方法而不是保存在数据库中，如此，你应该始终定期备份文件。

如果想保存到数据库中，那么需要能够实现保存到数据库中，因为 Builder不能自动做这个工作。

1) 通过选择Tools | Preferences菜单项显示Preferences对话框。

2) 在Access标签页上，找到左上角的 Access部分，选择单选按钮，从 File改为Database，或者把它设置为Ask，即在每次保存时均有Form Builder的提示(见图4-19)。

3) 现在选择File | Save菜单项保存所做的工作。

4) 如果选择File | Database，那么Form Builder弹出一个小的Filter的对话框，提示选择Database或File System。选择File System的按钮并单击OK。

Form Builder将工作一会儿，把表单模块存到数据库或文件中。用 Program | Run Form | Client Server菜单项，或Builder的垂直工具条上的Run工具，或在Layout Editor的水平工具条上的Run工具，都能够运行表单。 Builder编译表单，运行 Form Runtime程序，并运行表单。

China-pub.com 下载

图4-20显示了这个应用程序结果。

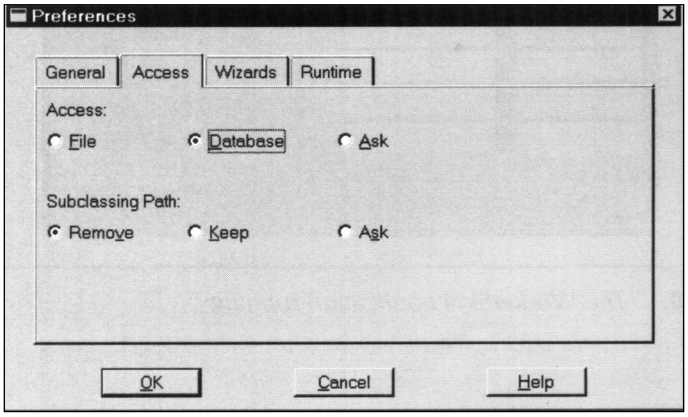


图4-19 在Preferences对话框中选择Access

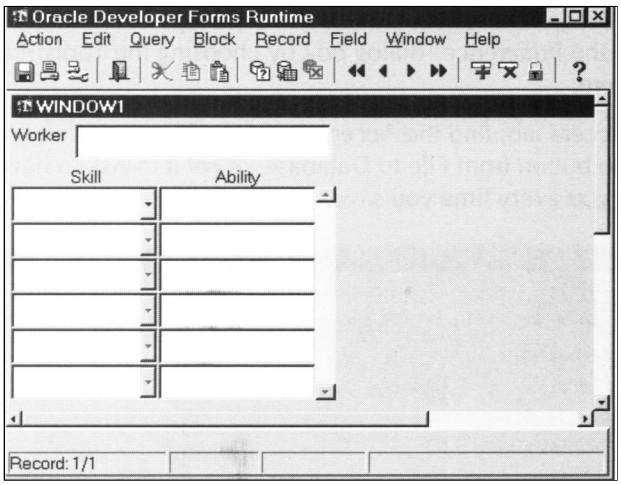


图4-20 运行WorkerSkill应用程序

注意 如果是为因特网开发这个应用程序，也可以利用Program | RUN Form | Web菜单项或与其相当的工具来运行这个在浏览器里的表单。要检验这个应用程序，可以试验某些操作。

5) 用菜单工具条上的Query Execute工具，查询所有的工人。

6) 使用Record | Next向下移动人员列表，注意是如何和何时得到有技能的人员的，这个表单自动地填写Skill和Ability域。

7) 在Skill域上单击，并下拉这个列表。

8) 选择一种技能，单击OK，填充Skill域。尝试键入一种原列表中没有的技能，发现输入不进去，这是因为Oracle Developer不理会键入的内容。

当完成了原形的测试时，关闭这个应用程序。对表单的任何改正要在 Form Builder中去完成。接下来还有几个步骤：把这个文件加到 Project Builder的工程层次结构中。

9) 打开Project Builder，在工程名上单击选择所创建的工程。

China-pub.com
下载

10) 单击工具条上的Add Files工具，并在所得的Add Files to Project对话框中找到Worker Skill.FMB文件。

11) 当在Open上单击添加这个文件时，Project Builder在工程中的Forms文档结点下为这个WorkerSkill.FMB文件加一个结点(见图4-21)。

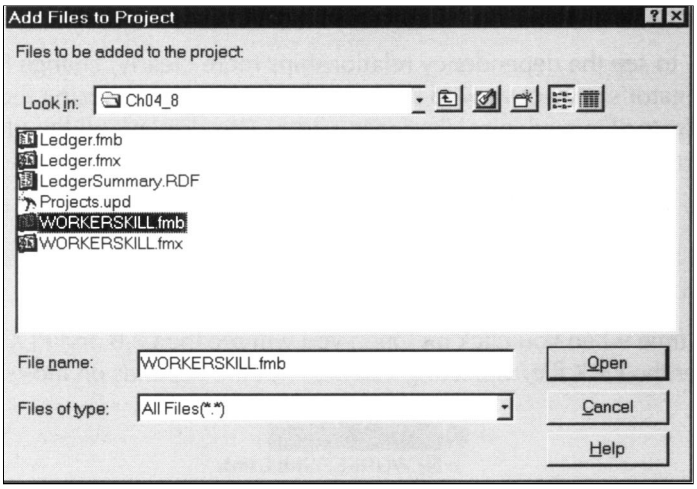


图4-21 为WorkerSkill.FMB文件加结点

Project Builder很聪明，在Forms的可执行的结点下加WorkerSkill.FMX文件。FMX文件是隐式项，它通常是编译或产生步骤的结果。当添加隐含项的输入项 (FMB文件)时，Project Builder自动地把隐含项加到工程中。当在Project Builder中用Build Incremental工具建立一个或多个逐渐增长的文件时，Project Builder理解这个隐含的关系，并重新编译过时的相关文件(见图4-22)。

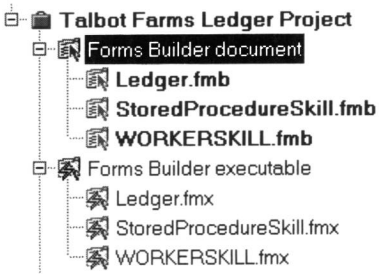


图4-22 Project Builder自动添加隐含项

虽然Project Builder可以推断许多对象之间的隐含的关系，但它看不到 PL/SQL代码中或对象库的从属关系。这就意味着当改变一个PL/SQL或对象库并重新编译所有的文件时，Project Builder并不自动地重新建立依赖于它们的应用程序。因此，在创建它们时，必须明确地输入这些关系。

12) 首先，为了比较清楚地得出从属关系，通过 Navigator |Dependency View菜单项，从Navigntor的工程视图变换到它的从属性视图，这将按从属关系显示所有的文件。

13) 现在选择FMX文件，并再在Add Files工具上单击。

14) 在Add Files to Proiect对话框中，找出TalbotStandard.OLB和TalbotStandard.PLX文件。

15) 再次在Open上单击，将看到OLB和PLX文件出现在FMX文件下面，说明FMX文件依赖于这些文件(见图4-23)。

图4-23 OLB和PLX文件出现在FMX文件下

用表单模块更新Project Builder，其从属关系让你在将来利用Project Builder的自动工具来管理这些文件。例如，如果打算使用配置管理工具，那么可以核对这些文件进入版本控制系统创建原形基准(有关细节请参看第15章)。然而，在修改这个表单原形之前，让我们去探索报表世界。

China-pub.com 下载

4.1.4 创建网格表单

在“创建主-从表单”的教程中使用单记录样式和多记录表格样式创建了单个表单。现在讲述多记录网格表单的一些专门技术，并为下一节分“使用 LOV向导”打下基础。

Ledger表单用网格形式提供 Ledger表(请参看附录A)。输入基本数据和查询 Ledger登记项将要使用这个表单。

1) 选择Object Navigator的Form结点，并在Add工具上单击，创建一个新表单，把表单的名字改为LEDGER，然后显示Property Palette，把Coordinate System的单位修改为英寸，就像在“创建主-从表单”部分为WorkerSkill表单所做的那样。

2) 选择Object Navigator的Data Blocks结点，并单击Add工具选择使用Data Block Wizard来创建新的数据块。

3) 在欢迎屏幕上单击 Next，这个数据块以数据块类型屏幕上的表为基础。在表屏幕上单击Browse按钮并选择LEDGER表，利用＞按钮把所有的基本列 (LEDGERID、ACTIONDATE、ACTION、ITEM、QUANTITY、QUANTITYTYPE、RATE和AMOUNT)移到这个数据块，还要把Person.Name域移到这个数据块，但是留下 Person REF域。Name替代Person reference，而且这个表单不要求任何其他的 Person列，单击Next(见图4-24)。

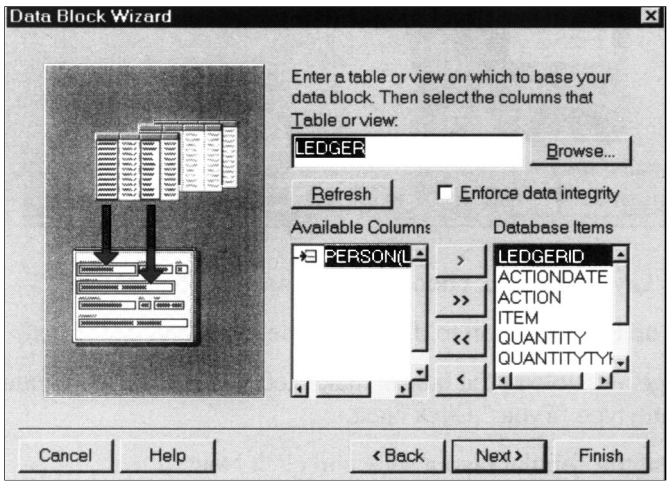


图4-24 表屏幕

注意 使用lookup columns是一个十分有用的特性，在Oracle 8中才有这个特性。在这里，表单不需要任何信息，需要做的全部工作是把涉及的列包含在数据块中，而 Form Builder将转交所需要的全部代码以连接导航到其他对象表。

Data Block Wizard显示一个屏幕，提问是否为 Person.Name项产生值的列表(LOV)。它知道如何做这个工作，因为此项目是一个查找。创建 LOV将使你能够通过从LOV中选择一个名字来更新数据库的REF列。

4) 检查Person复选框，然后在“...”按钮上单击，浏览数据库，查找 PERSON对象表。除非你有多个Person类型的对象表，否则只看到单个表。单击 Finish(见图4-25)。

5) 在Layout Wizard中，选择一个新画布并单击 Next。

6) 在>> 按钮上单击显示所有的项，并单击 Next。

China-pub.com
下载

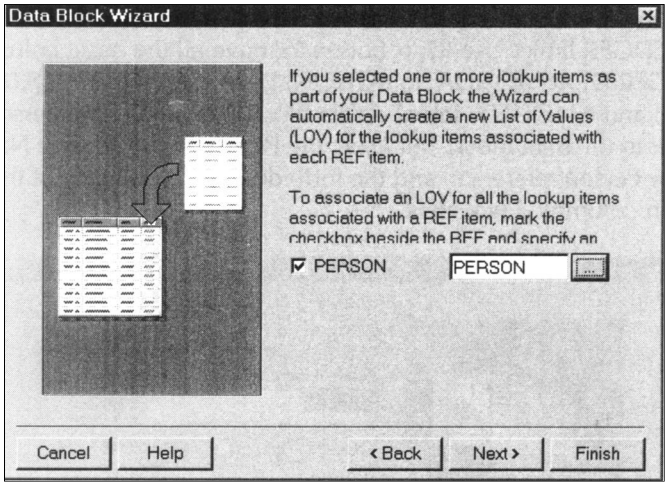


图4-25　查找PERSON对象表

7) 改变几个标记：使Ledgerid为“ ID ”、Actiondate为“ Date ”、Quantity Type为“ Type ”，单击Next。

8) 选择Tabular布局样式并单击Next。

9) 把记录的数目设置到 10，让Frame name空白，并查看复选框，显示滚动条，单击Finish。

Form Builder显示新的画布 —— 一个大的表格显示Ledger表的所有列(见图4-26)。

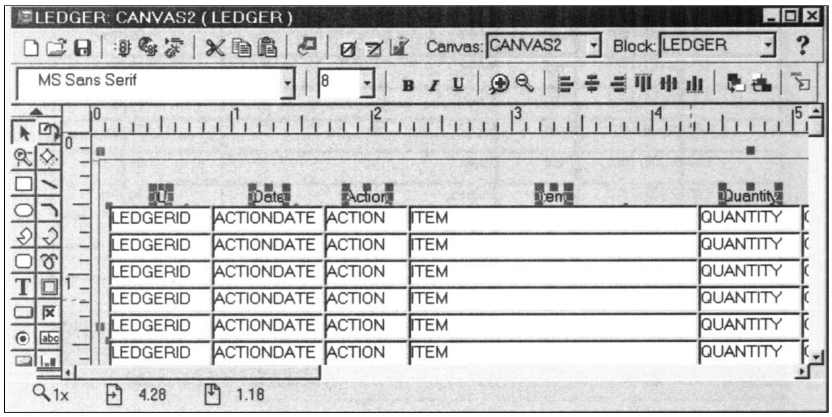


图4-26　Ledger表的所有列

接下来，我们需要把标准的SmartClasses应用到这些项中。

10) 进入Object Navigator，选择这个新的Ledger数据块下的每一项，并右击这个项，显示弹出菜单，选择SmartClass和适当的种类，比如Text。

注意　在查找数据块的项时，将看到Person在列表的顶上，Data Block Wizard创建了Person REF项和Person_Name项。如果查找Person项的属性，将看到一个Canvas(画布)属性的＜null＞值，意味着没有画布显示这个项。用户将通过Person_Name项和它的LOV更新这个项，在下面有更详细的描述。

China-pub.com 下载

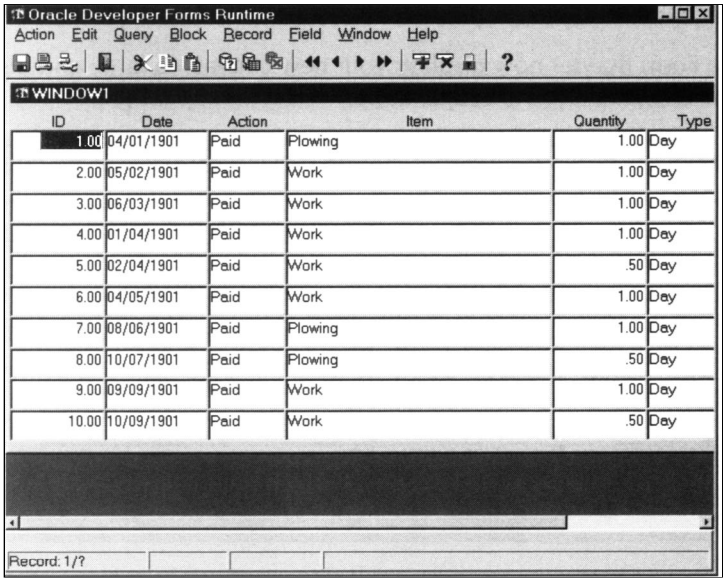


图4-27 Ledger表单

11) 如前所述，除去框架线条，把Frame尺寸属性设置为零，并把Shrinkwrap属性设置为Yes。

12) 运行这个表单，单击Execute Query工具进行查询，查看这些数据的显示模样。图4-27显示了结果。

13) 使用TAB键移到最后一项Person上，它显示人员的Name。如果查看状态条，会看到值列表的灯亮。如果选择Edit | Display List，则Form Runtime显示LOV，列出在Person表中所有的人名(见图4-28)。

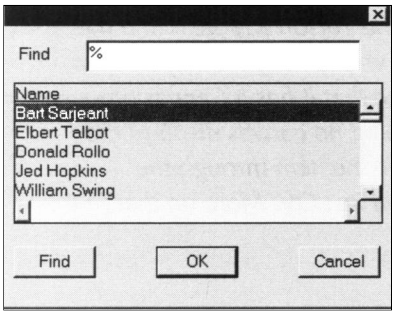


图4-28 显示Person表中所有的人名

4.1.5 使用LOV向导

当你使用从REF列导出的查找值时，Data Block Wizard可以容易地创建一个LOV。但是在没有REF的情况下，则要经常创建LOV。在前面“Skill列表”部分中，已经通过创建下拉列表的步骤工作过了。当显示项数少于10或15的有限的项目时，采用这些列表是不错的。在项数多于15的情况下，可以利用LOV的特性：滚动条和模式匹配。Person LOV是一个利用这种特性得到好处的完美例子。

采用下拉列表，可以键入选项的第一个字母，列表把这个选项定位在用这个字母开头的第一项上，如果再键入几个字母，它就走到用这些字母开头的那一项。在LOV中键入多个字母，LOV就一次性把选择定位在正确的项上。可以利用Ledger Person LOV来试验一下，看它是如何工作的。

下面是LOV Wizard教程，这是一个工具，用来更容易地为项目创建LOV。本教程给你以下几个技能：

- 如何利用LOV Wizard创建LOV。
- 如何规定LOV的列表项。

China-pub.com
下载

■ 如何把LOV选择的结果与数据块项相联系。

■ 如何为LOV创建标题。

■ 如何利用LOV确认数据块。

教程中增加一个LOV，将为Quantity Type项提供值，数量类型经常在数据库中扩展，所以，Talbot不希望在这里只提供下拉列表。

1) 选择Tools | LOV Wizard菜单项，LOV Wizard启动，并询问LOV记录组是基于查询还是基于现有的记录组，选择前者并单击Next(见图4-29)。

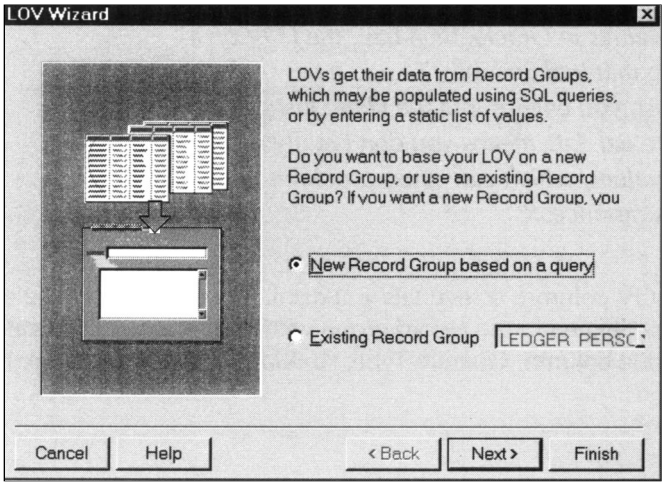


图4-29 启动Lov Wizard

2) 把下面这个SQL语句输入到查询屏幕中，并单击Next：

```
SELECT DISTINCT Quantity Type
  FROM  Ledger
ORDER BY 1
```

DISTINCT限定词告诉Oracle从Ledger表只返回一组性质不同的值。LOV将显示表内当前的所有值。ORDER BY规定以字母数字顺序显示这些值，这是显示LOV的值的常用方法(见图4-30)。

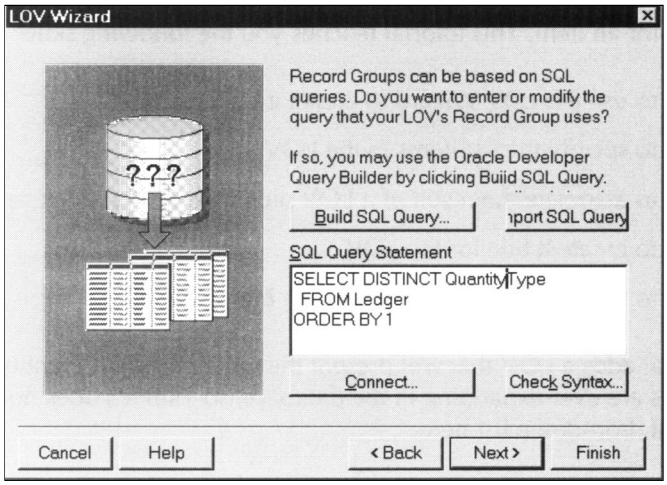


图4-30 显示LOV值的常用方法

China-pub.com 下载

提示 通常希望对这组值有稍微多一点的控制。在Oracle中可以创建单独的表来包含这些值，然后以LOV为基础从那个查找的表中查询。要加一个新值到LOV列表中，就把它输入进这个表。如果只能从实际的数据得到这组值则意味着不能真正控制它。在某些环境下还可能会引起问题。

3) LOV列的屏幕用查询来显示在记录组定义的一个或多个列。在这里只有一列 Quantity Type，双击显示它。单击Next(见图4-31)。

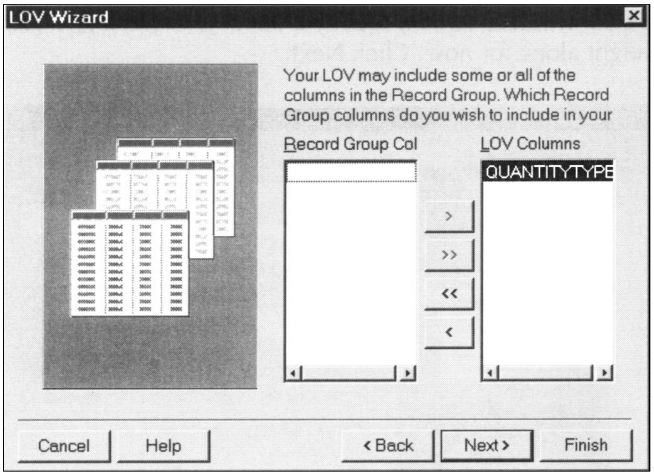


图4-31 双击QuantityType列

4) 在属性屏幕中，指定这个项目的列标题 —— Type。单击屏幕的底部 (是单击“ look up return items ”标签，它不是总可以看得见的)。从摘选列表中选择 Ledger.QuantityType项并单击OK。单击Next(见图4-32)。

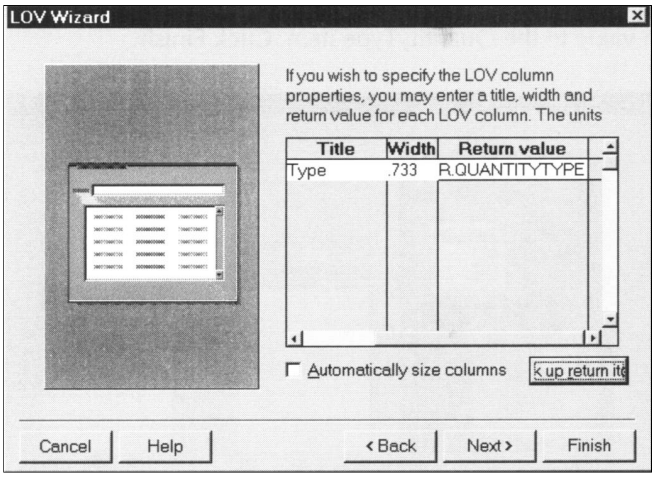


图4-32 选择Ledger.QuantityType项

用专门的数据块和项设置与列表值有关的返回类型。接下来的屏幕将使用这个值，所以在这里设置它是很重要的。

5) 在LOV窗口屏幕中，指定标题： QuantityType。不改变其中的宽度和高度，单击 Next (见图4-33)。

China-pub.com
下载

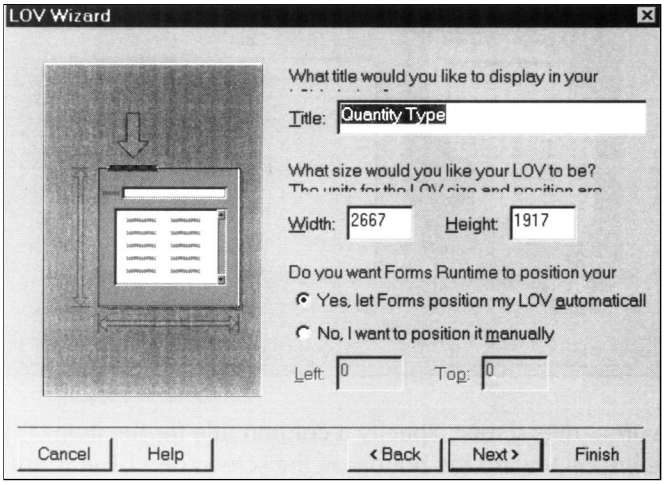


图4-33 在LOV窗口屏幕上指定标题QuantityType

6) 单击Next，接受所有的高级属性。然后在 Assigned Items屏幕中选择QuantityType项作为指定的项。这把返回值分配给 QuantityType项，单击Finish(见图4-34)。

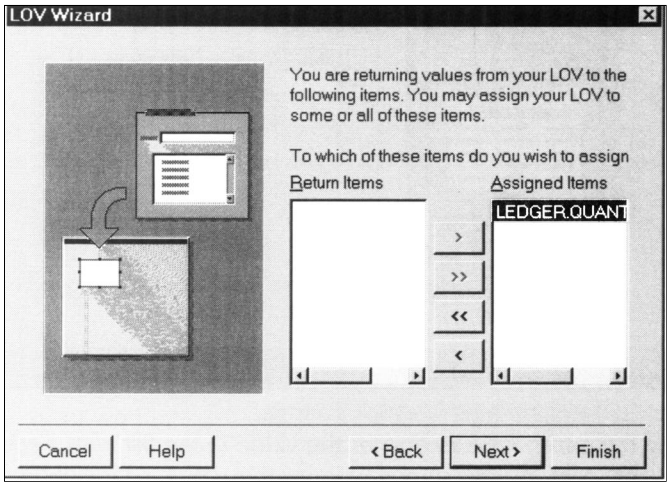


图4-34 在Assigned Items屏幕中选择QuantityType项

7) 在Object Navigator中找出这个新的LOV，重新命名它为“ QuantityType ”，重新命名相应的记录组为“ Quantity Type ”。

8) 导航到Ledger.QuantityType项，显示它的Property Palette。找出值列表(LOV)，把List属性的Validate设置为Yes(见图4-35)。

- List of Values (LOV)	
List of Values	QUANTITYTYPE
List X Position	0
List Y Position	0
Validate from List	Yes

图4-35 找出值列表

China-pub.com 下载

最后一步告诉Form Runtime使用LOV值作为一组值，用这组值去验证输入到域中的数据。因此用户只需要输入列表值。在这里，意味着用户输入的值限制为数据库的当前值。

9) 运行表单。单击Execute Query按钮和Type项上的标签，使用Edit | Display List来显示LOV(见图4-36)。

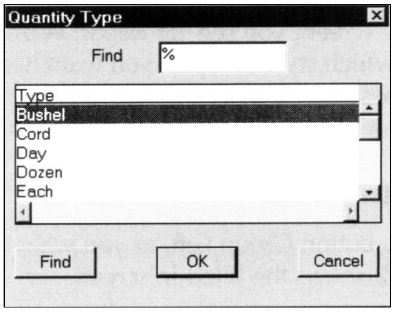


图4-36 单击Execute Query按钮和Type项上的标签

4.2 开发简单汇总报表

报表的建立像表单一样容易，但它们完成的任务不同。建立报表的许多任务与报表布局的格式有关，与构造对象所做的事情相反。

表格报表根据单个SQL SELECT语句打印出表。汇总表格报表计算部分和并实现其他的归并操作。在建立Talbot Ledger Summary报表过程中，讲述如何迅速地产生汇总报表——一种极普通的报表。

单击Project Builder Launchpad上的工具，启动Report Builder，Welcome(欢迎)对话框询问是否要运行Report Wizard。利用Wizard来定义汇总报表是一个非常简单的过程，单击单选按钮Use the Report Wizard，启动Wizard。

在下面的教程中，你将学习以下技能：

- 如何用Report Wizard建立一个简单的汇总报表。
- 如何把数据库列映射到报表的域。
- 如何产生单个组和汇总列。
- 如何利用标准的报表样板来格式化基本的报表。
- 如何利用Live Previewer重新布置缺省的报表布局。
- 如何在Live Previewer中给报表加标题。

Ledger Summary报表显示Ledger表的数据，这些数据按照人员分类，带有活动日期、项目说明以及收入支出的总数。在这个 Ledger中，每个人员有一个小计，表示这个人的买卖总量。

4.2.1 运行Report Wizard

在Wizard的欢迎屏幕以后，看到Report Wizard样式屏幕，如图4-37所示。在这里选择想要的报表的类型(有关不同类型的报表的详细内容，请参看第7章)。Ledger Summary报表是一个Group Left 报表，即在报表的左边有一组记录。

1) 输入报表的标题“ Ledger Summary Report ”。

2) 因为想要一份分组的汇总报表，单击单选按钮 Group Left。图4-37显示了满屏的屏幕。

3) 在Next上单击。

下一屏提示是使用SQL语句还是使用Express查询。如果安装了Oracle Express,这个数据挖掘工具在开发查询中可能非常有效。本教程使用 SQL语句。

4) 选择SQL Statement单选按钮，单击Next。

下一屏提示输入基于这个查询的SQL语句，如图4-38所示。输入以下查询：

China-pub.com
下载


```
SELECT p.Name, l.ActionDate, l.Item,
       l.Quantity, l.Rate, l.Amount
  FROM Ledger l, Person p
 WHERE l.Person = REF(p)
```

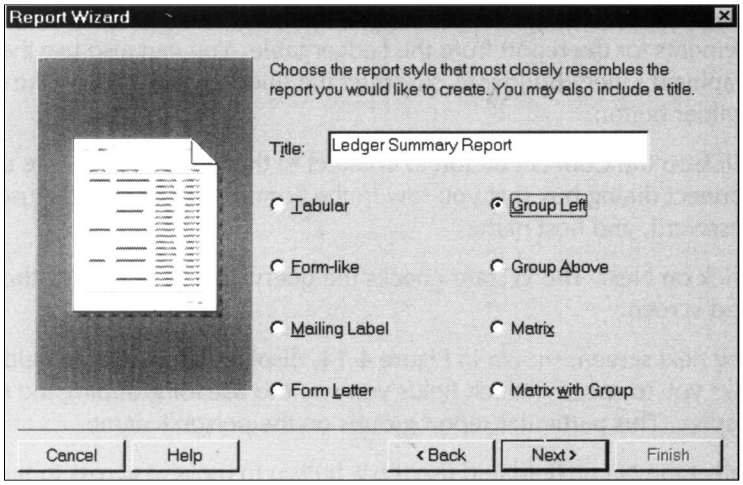


图4-37 在Report Wizard报表样式屏幕上设置Group Left报表

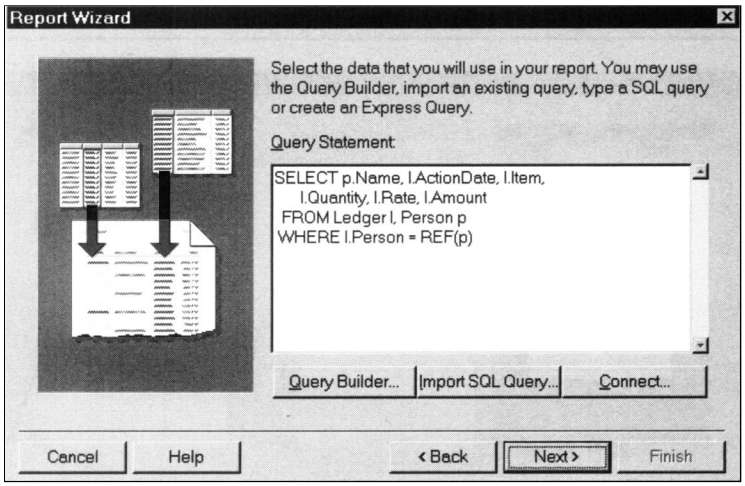


图4-38 Report Wizard使用Ledger Summary查询的SQL查询屏幕

这个查询利用Person REF列连接Ledger和Person表，从Person表返回人员名，并从Ledger表为报表返回另外五个数据元素。也可以利用图形的 Query Builder，通过单击Query Builder按钮来构造这个查询。

5) 单击Connect(连接)按钮，连接数据库。可以看见与 Form Builder中看到的一样的连接对话框。输入用户名、口令和主机名。

6) 单击Next，Wizard检查这个查询，然后进到下一个屏，如图 4-39所示，显示现有的域并要求指定你想用哪些域来对报表显示进行分组。这个报表是根据人员的名字分组。

7) 选择Name域，并利用＞按钮把它移到Group Field列表中，单击Next。

下一屏如图4-40所示，显示现有的报表，并提示指定以什么样的次序显示哪些域。不需

China-pub.com 下载

要显示所有的项，只要显示想看到其值的那些项，其他项可能在报表中用来作计算域的基础或者为了另外一些用途。

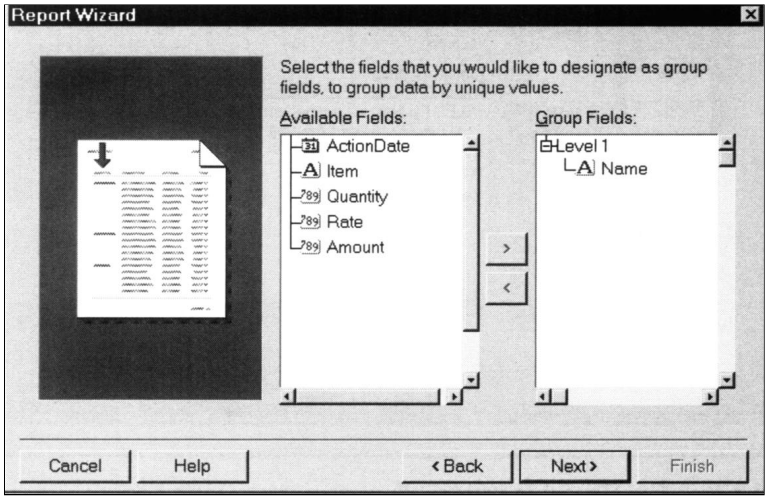


图4-39 Report Wizard中带有Person分组的分组域屏幕

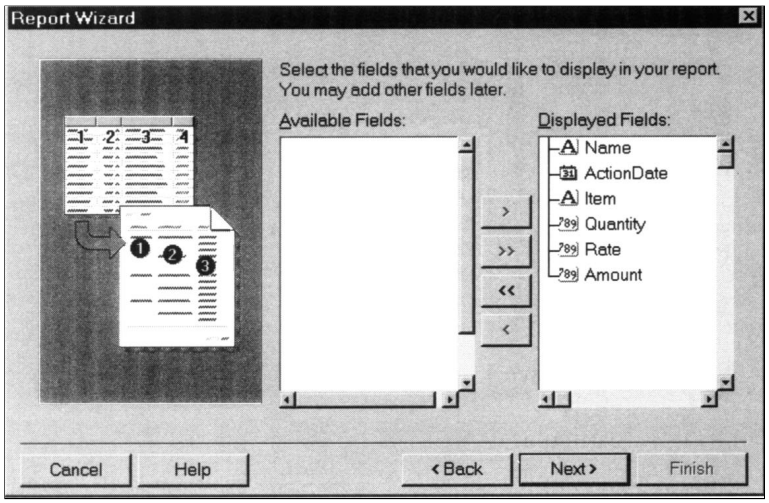


图4-40 Report Wizard中带有Ledger Summary域的域屏幕

8) 单击>>按钮，把在Available Fields列表下的所有的项移到Displayed Fields列表中，单击Next。

现在可以看见总计屏幕，为使用总计域选择一个总计函数（求和、平均、计数、最小值、最大值或总计百分比）。Ledger Summary报表对每个人员求分类汇总。

9) 选择Amount域并单击Sum＞按钮。Wizard为Amount域创建一个求和总计，如图4-41所示。单击Next。

下一屏列出这些域以及它们的标记和宽度

10) 把ActionDate改为“ Date ”。把第二个Total改为“ Grand Total ”。图4-42显示了改变了的域名。单击Next。

China-pub.com
下载

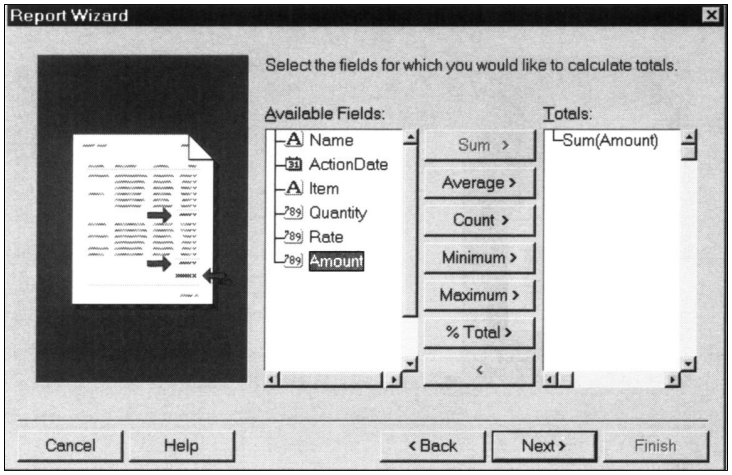


图4-41　Report Wizard中带求和域的总计屏幕

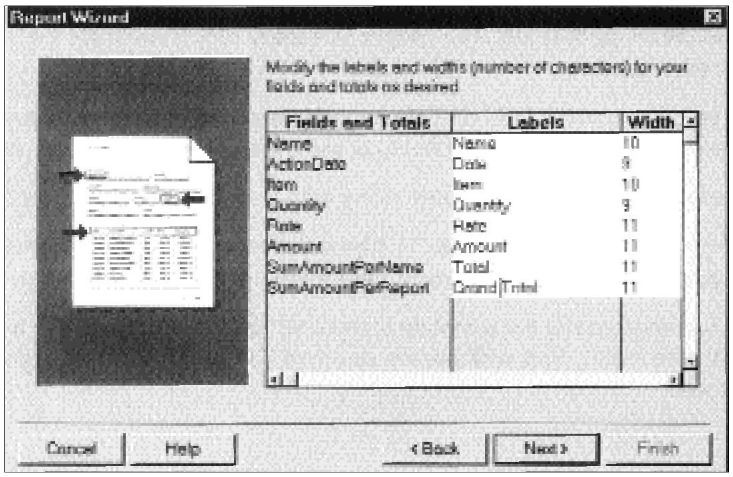


图4-42　Report Wizard的域名屏幕

这将进入模板屏幕，它让你把模板和报表联系起来，使用标准的Talbot模板，如图4-43所示。

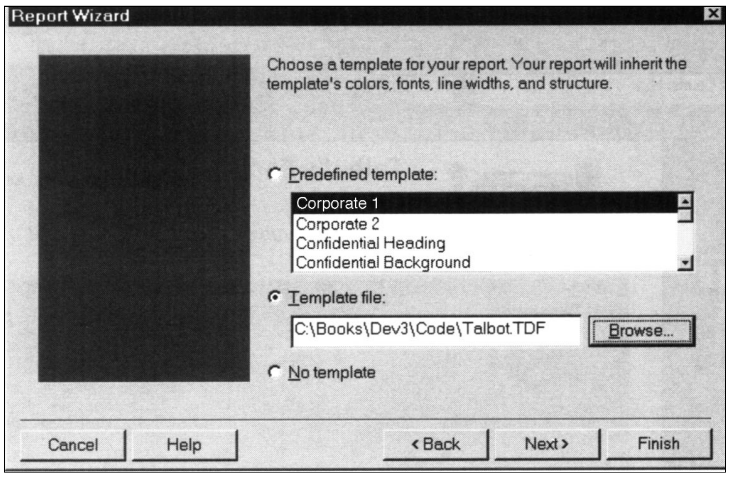


图4-43　带有Talbot模板的Report Builder模板屏幕

China-pub.com 下载

注意 你需要从教程目录的各种文件中定位这个标准的Talbot模板文件Talbot.tdf。可以利用靠近Template文件域的浏览按钮找到它。

11) 选择Template文件单选按钮，然后通过使用 Browse按钮选择Talbot.tdf文件填入Template文件域，单击Finish，创建这个报表。

现在Report Wizard生成了报表，并显示带有这个报表的 Live Previewer，如图4-44所示。虽然这里讲的是一些基本的东西，你可能想改进报表的外观，并使安它提供数据更加有效，你可以用Report Wizard或Live Previewer完成其中的大部分工作。

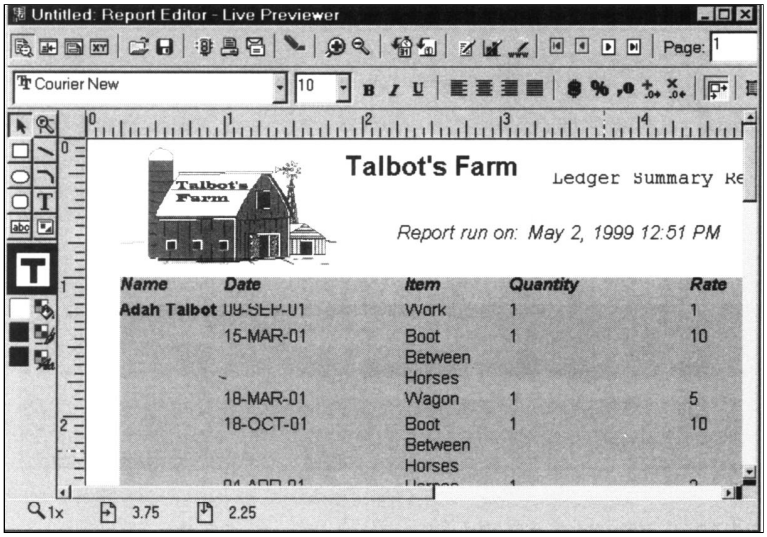


图4-44 带有初始Ledger Summary报表的Live Previewer

4.2.2 使用Wizard之后的整理

为了把报表改进为结构好的报表，可以使用 Live Previewer来改进格式：

- 改进域的大小并重新布置它们，使其占据比较少的空间。当它们的位置在页面上太分散或域的尺寸太短时，可能会在域中引起不必要的卷绕。
- 格式化货币域。
- 给报表加标题。

要决定域的大小，可以在 Live Previewer的各处移动它们。 Live Previewer的缺省模式为Flex(伸缩)模式。这个模式把域限制在页面的边缘的范围内，所以你可以把域向上移到离开页面的地方。用Previewer工具条的Flex Mode工具可以关闭这个模式。例如，如果右边的页面有空间，那么当拉一个域到右边时，在那个域和右边缘之间移动以容纳这个加宽的域。如果没有空间，那么不能加宽这个域。这种布置像在做游戏。必须反复安排组件，前后移动它，直到把它们布置为所希望的式样。

例如，为了加宽Person域，必须减小数目域的大小，在其右边部署一些空间。

1) 单击那列的值，选择Quantity域。

2) 单击这列右边的手柄，并把它拉回到减小的域的尺寸数。

到处可以看到手柄，通过在手柄的任一边单击可以把域拉得比较宽。如果要到这个域不能到的某个地方，比如对页面来说太宽了，那么鼠标光标将转为一个“ Not ”符号。这需要体

China-pub.com
下载

验才能够完全理解。所以，应该试一试。通常重新定域的大小为了适应数据和标题的大小。对于Quantity域，其数字是单个或双个数字，而且标题“ Quantity ”太长了。

3) 单击选择“ Quantity ”标题，在这个标题上双击激活文本编辑，然后把这个标题改为“ Qty ”。这个新标题小得多，但比在它下面的数字长。

4) 再重新确定Quantity域的大小，使它更小些。

5) 对于Rate和Amount域，重复以上确定尺寸的过程，但不需要改变标题。

6) 将这些域尽可能拉到右边，然后再把 Date和Item拉到右边。

7) 选择Person域。然后在手柄上单击，并把边界拉到右边，直到它有大约 1.75英寸长。这一列没有名字卷绕。

8) 为了格式化货币域，选择这个域，然后在 Live Previewer工具条上的格式工具上单击(见图4-45)。

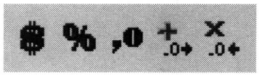


图4-45 单击Live Previewer工具条上的格式工具

用$工具加一个货币符号，用逗号工具加一个数字分隔符号，用plus工具加一个小数位。不要忘记格式化 total 域以及 Amount和Rate域，包括在报表末端的总计。

9) 选择一个域，然后单击Justify Right 工具，使得三个数字域的两端对齐。

报表的标题按Courier类型出现在上边缘的中央。需要修改它的位置和字型，使它和报表的其余部分一致。

① 在标题上单击，选择它。

② 使用工具条上的字型和尺寸下拉列表，把字型设置为 Arial18粗体或某些类似的有吸引力的字型。

③ 使用尺寸手柄重新确定域的大小直到整个文本显示精美为止。

④ 重新布置报表头部的各种元素，用吸引人的方法显示标题。

提示 Live Previewer的缺省设置是用Destination Type Previewer(目标类型预览器)显示表单。用这种类型显示Previewer中的打印文本稍微有些不对，它切断了一些文本。可以通过把Destination Type设置为Screen来解决这个问题。进入 Object Navigator并找到 Reports | UNTITLED | System Parameters |DESTYPE对象，双击这个对象显示它的 Property Palette。找出Initial Value 属性，通过下拉列表把它设置为Screen。用Run工具运行这个报表，重新显示在Previewer中的报表。

4.2.3 保存和运行报表

现在是保存报表的好时机。使用 Tools | Preferences菜单项访问Preferences对话框和它的Access标签页，选择报表的保存处 (文件、数据库或文件 /数据库)。Builder使用你提供的文件名作为这个报表的名字。当单击 Data Editor工具条上的Save工具时，Builder保存这个报表的各个部分，提示需要文件名，并用你给报表的名字 (在这里是 Ledger Summary. RDF)更新 Object Navigator。

你可以运行这个报表，观看它的模样。在 Run按钮上单击，显示Live Previewer。

4.2.4 为Web显示创建报表

此时，可以用HTML和Adobe Acrobat格式产生一个Web报表。观看这个报表联机时的模样。

China-pub.com 下载

1. 创建HTML报表

Hypertext Markup Language(HTML) 是SGML标准的标记语言的扩展，用来格式化 Web浏览器的文档(文件)。HTML为组织有效的Web文档提供了许多特性。包括超级文本连接和运行各种应用程序的有效连接。因为 HTML是运行在任何Web浏览器的标准语言，所以可以在任何运行Web浏览器的平台上运行HTML页面。因此，对于以一种与平台无关而且容易使用的格式提供报表来说，HTML是理想的工具。

下面的教程给你这样一些技能：

■ 如何使用Web Wizard产生HTML报表。

■ 如何将HTML书签加到报表中。

■ 如何在报表中添加标准的HTML页眉和脚注。

1) 在Web Wizard工具上单击，你看到欢迎屏幕，然后是书签屏幕。

书签屏幕让你在报表输出中产生一个书签的轮廓。书签是显示在报表边上的字符串。当在书签上单击时，浏览器将它放在报表相应的位置。 Web Wizard让你使用任一组域作为书签生成器。

2) 利用 > 按钮使Name组域作为书签生成器，如图 4-46所示。

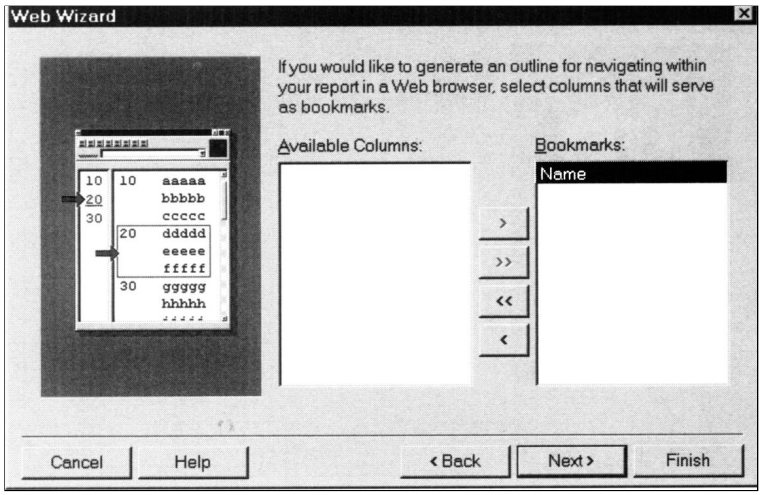


图4-46 Web Wizard书签屏幕

下一个屏让你在报表的开头和末尾包含文件的标签，可以利用这个特性，在报表 Web显示的页前或页后显示页眉和脚注或任何其他的 HTML格式。

3) 这个报表不使用任何另外的HTML所以单击Next。

进入最后一屏，如图 4-47所示。它让你生成 HTML页面、HTML风格的图表、 Acrobat PDF，或者什么也不产生。通过选中“ Generate output to a Web browser ”复选框，可以在缺省Web浏览器上显示HTML页面或PDF文档(文件)。

4) 选中“ Generate output to a Web browser ”复选框，在“ Generate HTML output now ”单选按钮上单击，产生一个HTML文件。

5) 在Finish上单击，产生报表。把HTML文件存到磁盘。图4-48显示了在Netscape Web浏览器上的这个报表。

China-pub.com
下载

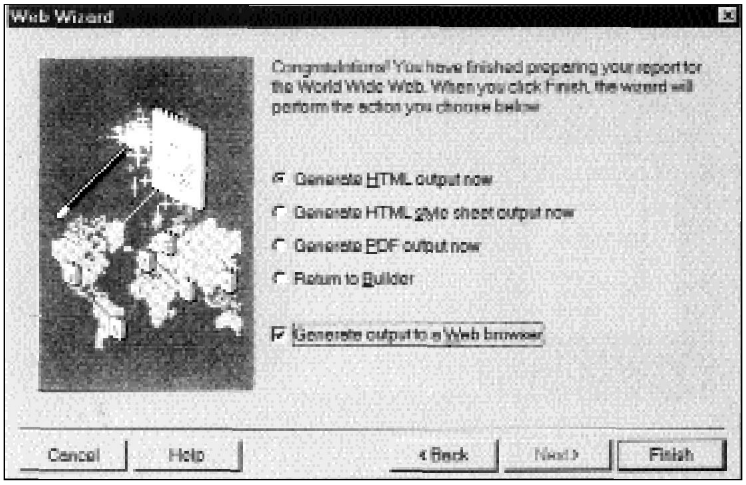


图4-47 Web Wizard的生成屏幕

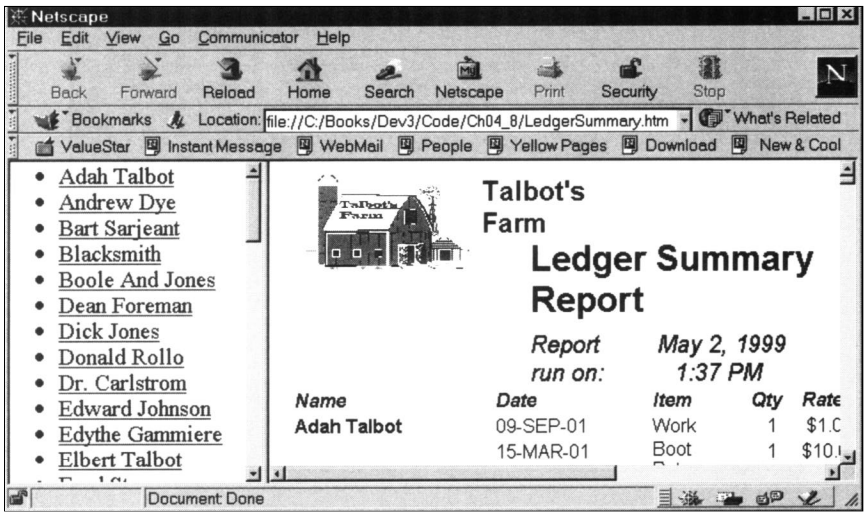


图4-48 使用HTML的Ledger Summary 报表原形

2. 创建Adobe Acrobat报表

Adobe Acrobat是Adobe Postscript语言的扩展，让你用联机的形式来表示文件，就像它们出现在硬拷贝那样。Acrobat给文件增加了超级文本和 table-of-contents (bookmark —— 书签)的能力，让它成为联机文件的有效工具。还可以在任何有 Acrobat Reader(大多数都有)的操作系统或用户界面中显示Acrobat文件。因为Oracle Developer的报表是静态文件，所以对用一种与平台无关并容易使用的格式提供报表来说， Acrobat是理想的工具。

下面的教程教你以下的技能：

■ 如何使用Web Wizard生成Adobe Acrobat报表。

■ 如何定义自动生成书签。

1) 在Web Wizard工具上单击，将看到欢迎屏幕，然后是书签屏幕。

书签屏幕让你在报表输出中制作一个书签轮廓， Acrobat书签是一个标签，呈现在 Acrobat的专门的书签窗格中。Acrobat把这个窗格显示在Acrobat窗口文件窗格的左边，给出一个内容

China-pub.com 下载

的交互表。当在书签上单击时， Acrobat将其放在报表的相应位置上。 Web Wizard让使用任何组域作为书签产生器。

2) 利用>按钮使用Name组域作为书签产生器，如图 4-49所示。下一屏让你在报表的开头和末尾包含文件的HTML标签。可以利用这个特性在报表在 Web页面显示前或显示后显示页眉和脚注或任何其他的HTML格式。

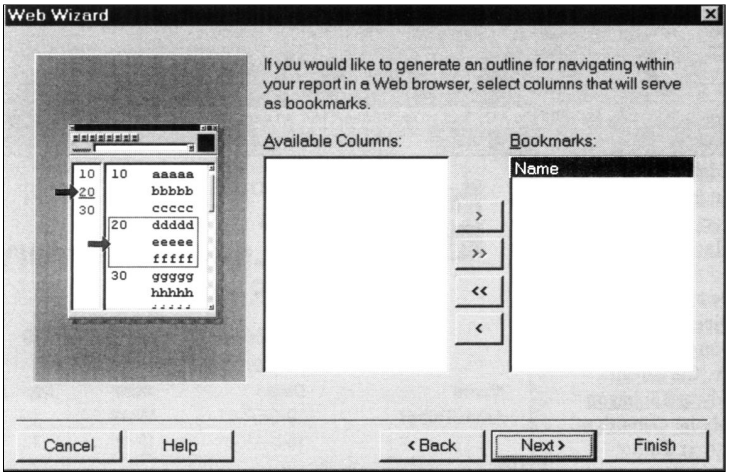


图4-49 Web Wizard书签屏幕

3) 因为产生一个PDF文件，而不是HTML文件，所以单击Next。

这样就进入到了最后一屏，如图 4-50所示。你可以生成 HTML页面、 HTML样式的表格、Acrobat PDF、或什么也不产生。通过选中“ Generate Output to a web browser ”复选框，可以在缺省Web浏览器中显示HTML页面或PDF文件。

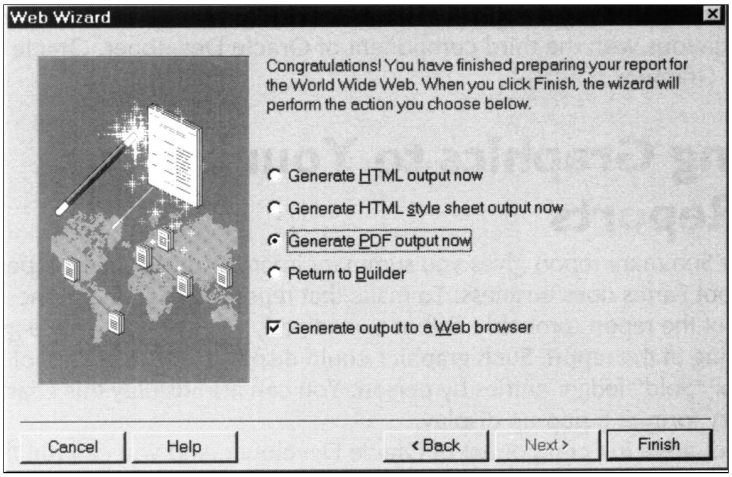


图4-50 Acrobat的Web Wizard生成屏幕

4) 选中“ Generate output to a Web browser ”复选框，在“ Generate PDT out now ”单选按钮上单击，产生一个Acrobat文件。

5) 在Finish上单击，产生报表。把这个 PDF文件存到磁盘上。图 4-51显示了运行Acrobat Exchange plug-in的Netscape Web浏览器的报表。

China-pub.com
下载

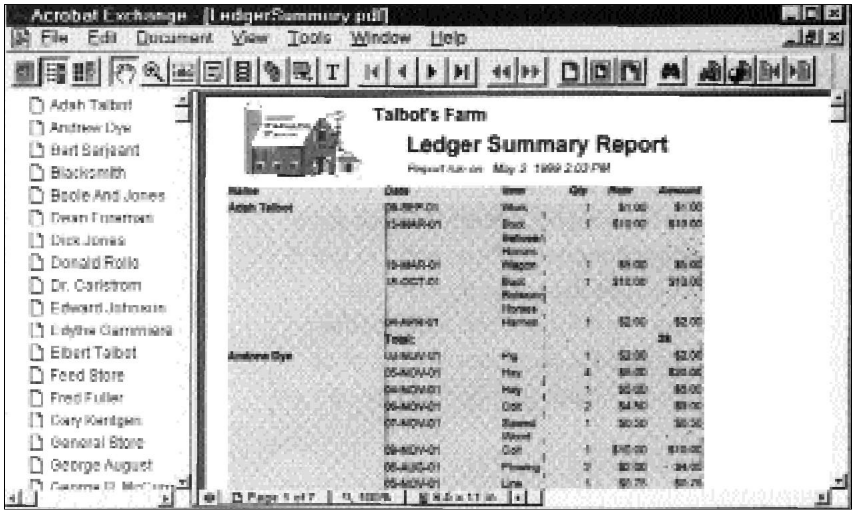


图4-51　Acrobat形式的Ledger Summary报表原形

注意　在试用产生和显示Acrobat文件之前，应使Web浏览器运行，否则，你将得到一个错误信息说程序找不到浏览器。

几个部分的教程给你示范了表单和报表的基本特性。现在，该考虑如何用 Oracle Developer、Oracle Developer Graphics (Oracle开发器的图形) 给你的应用程序添加一些色彩了。

4.3　为表单和报表添加图形

Ledger Summary报表给出了和Talbot农场做生意的每个人的概要信息，为了使这个报表对报表的用户(可能是Talbot各级经理)更有用，你可以在报表的开头加一些简单的图形，这样的图形能够根据人员显示Ledger的“ Bought ”和“ Sold ”两个登记项。还可以在Ledger Entry表单中以弹出方式显示这个图表。

图形是Oracle Developer的一个丰富的组件。如果需要的话，可以单独地运行它们。可以通过Oracle Developer Graphics Runtime模块(Oracle Developer图形运行时模块)使图形作为可独立使用的应用程序运行。可以通过 Web浏览器使它们有效，浏览器自动地让用户从可以利用的图形中进行选择。对大多数应用程序来说，可能希望把图形和图表嵌入表单和报表中，这正是这一部分要讲的内容。除非想建立一个独立的图形显示模块为以后几个地方再使用，否则不必单独地运行 Graphics Designer(图形设计器)，而可以使用 Form Builder或Report Builder中的Chart Wizard(图表向导)。

注意　图形显示可能从嵌套的报表或表单中或通过单独的查询直接从数据库中得到数据，Chart Wizard假设你使用已经在表单或报表的数据来产生图表。如果想使用SELECT语句直接从数据库中得到数据，那么由scratch而不是利用Wizard建立图表。或者建立单独的数据或数据模块，图形显示可以从它们那里得到数据。

4.3.1　在表单中创建图形显示

在把一个图表嵌入表单时，通常希望这个图表补充和使用表单的数据。例如在 Ledger表

China-pub.com 下载

单中，可以加一个折线图来表示随时间变化的数量，或者用一个饼形图来显示总量的分布情况。如果使用表单的通过例子查询的特性，那么会希望这个图反映所选择的数据的子集。强有力的工具可以使你只用图形表示想看到的信息。

这部分教程教给你以下的技能：

- 如何在表单中创建图形显示。
- 如何使用Form Builder Chart Wizard。
- 如何使用数据块的数据作为图形显示数据。
- 如何使用从数据库集成的数据作为图形显示的数据。
- 如何在画布中确定图形显示的数据和位置。
- 如何通过按钮确定图形在指定窗口中显示的位置。
- 如何导航到一个活动而隐藏的窗口中。
- 如何把图形显示作为一个单独的可再使用的模块来保存。

这部分教程教你在“创建网格表单”部分中开发的 Ledger应用程序中建立两个图形。第一个图形是折线图，用曲线表示在一段时间内事务的数量。例如图形可以显示由于不平常事务而朝向比较大的事务或临时的尖峰信号的某种趋势。这个图形将出现在 Ledger登录项滚动表的下面，并从那个表中提取它的数据。第二个图形是饼形图，用图表按照类型表示数量的分布，显示对于总数来说有多少销售和购买的事务数。这个图形出现在单独的窗口，通过单击Ledger数据数上面的按钮，一经请求就可以访问这个窗口。

注意 教程假设你已经熟悉了建立数据块的基本过程，所以把数据块建立过程缩得很短。如果没有完全熟悉数据块的建立过程，请学习Form Builder教程。

1) 打开Ledger表单，在Object Navigator中选择单击Data Blocks结点，然后在Add工具加一个名为CHART的新数据块，选择人工建立数据块方式，然后在缺省的名字上单击，将它改为CHART。

2) 设CHART块为控制块。打开这个数据块的 Property Palette，找到Database Data Block属性，并把它设置为NO。

3) 在Object Navigator中找出画布，并在其上双击，在 Layout Editor中显示这个画布，拉出画布在Ledger网格下，留出一些空间把图形项安置在这些空间中。

4) 在Layout Editor的Chart Item工具上单击，在已经存在的包含 Ledger表的网格下画一个长方形。

5) Oracle Developer询问是否使用Chart Wizard，单击OK(见图4-52)。

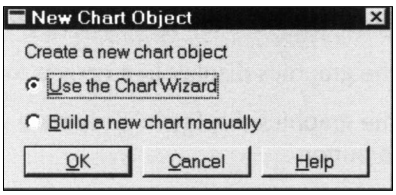


图4-52 使用Chart Wizard

下一屏让你输入这个图形的标题并选择你要什么种类的显示。第一个列表包含有不同种类的图形。第二个列表的内容根据第一个列表中的选择而变化。第二个列表的每一个元素表示在第一个列表中选择的图形类型的子类型。

6) 输入标题“ Amount Trend ”，在图形类型屏幕选择 Line图和Plain子类型，然后单击Next(见图4-53)。

为了确定图形的数据，首先通过 Chart Wizard的数据块屏幕选择一个数据块作为数据源，

China-pub.com
下载

如图4-54所示。

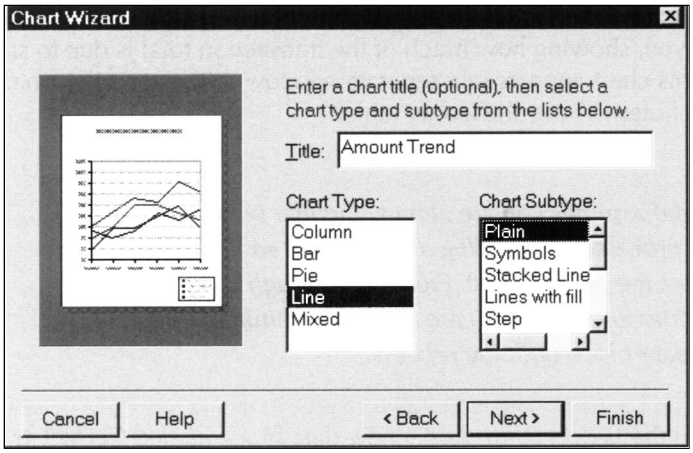


图4-53　在图表类型屏幕选择Line图和Plain子类型

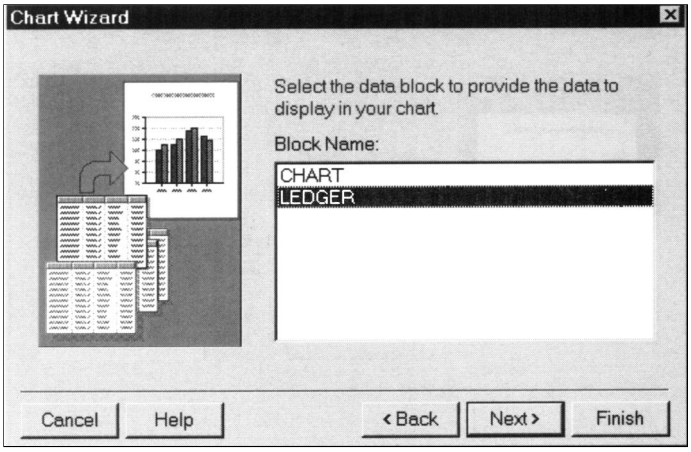


图4-54　Forms Chart Wizard数据块屏幕

7) 选择Ledger屏单击Next。Wizard显示种类子屏幕或X-轴子屏幕。如图4-55所示。

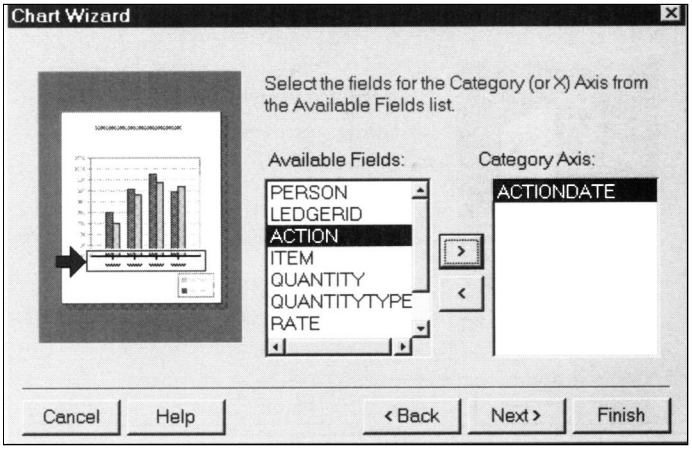


图4-55　Chart Wizard种类屏幕

China-pub.com 下载

8) 选择ActionDate域作为X轴——时间轴，单击Next，Wizard显示值的子屏幕或Y轴子屏幕，如图4-56所示。

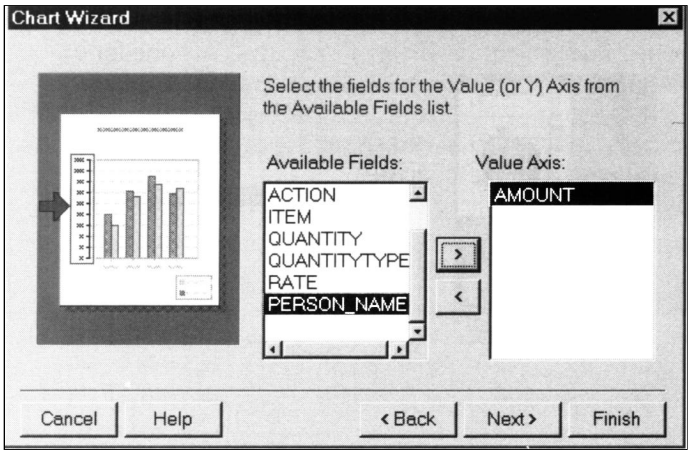


图4-56 Chart Wizard的值屏幕

9) 选择Amount作为Y轴——为每个日期显示的数量。单击Next。图4-57显示了最后一屏。

10) 在Finish上单击，把这个图形作为AmountTrend . OGD保存。

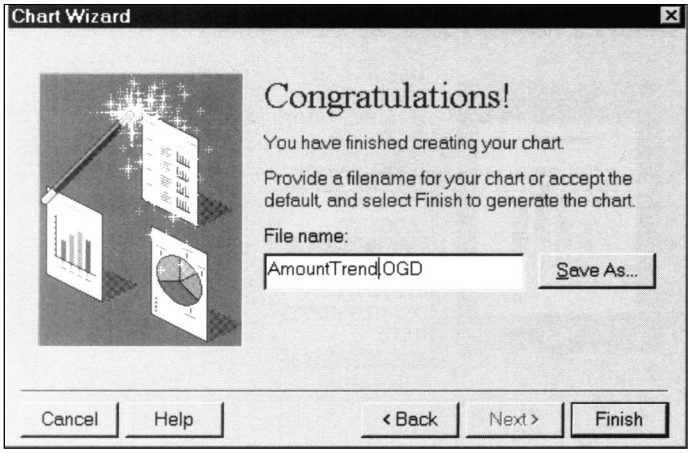


图4-57 Chart Wizard保存屏幕

现在已经把折线图嵌入了Ledger表单。可惜的是，它是Ledger数据块的一个项。如果在画布上向下滚，那么你重复看到这个图形，那是因为 Ledger表格的每一个记录都有一个图形。为了使它只显示一次，需要把这个图形移到CHART数据块。

11) 重新命名这个图形项为“ AMOUNTTREND ”。把它拉到CHART数据块中。

如果用Run工具运行这个表单，并用Execute Query工具在Ledger块中执行查询，你将看到用这些数据值填充的图形，如图4-58所示。

现在创建根据活动分解的饼形图，在一个独立的窗口显示出来。这部分教程要求从数据库中取数据而不是从Ledger数据块取数据。就是说，这个图形汇集的有关 Ledger事务的信息不使用Ledger数据块的原始数据，而你希望根据活动将这些值分组并显示每组的和。因为Ledger数据块中没有与这些数据对应的计算域，而且因为数据块的记录和要在图形上显示的

China-pub.com
下载

项不是一一对应的，所以必须单独查询这些数据。这就要求一个单独的数据块保存这些汇集的数据，加上一些代码以便一经请求就得到数据并显示这个图形。

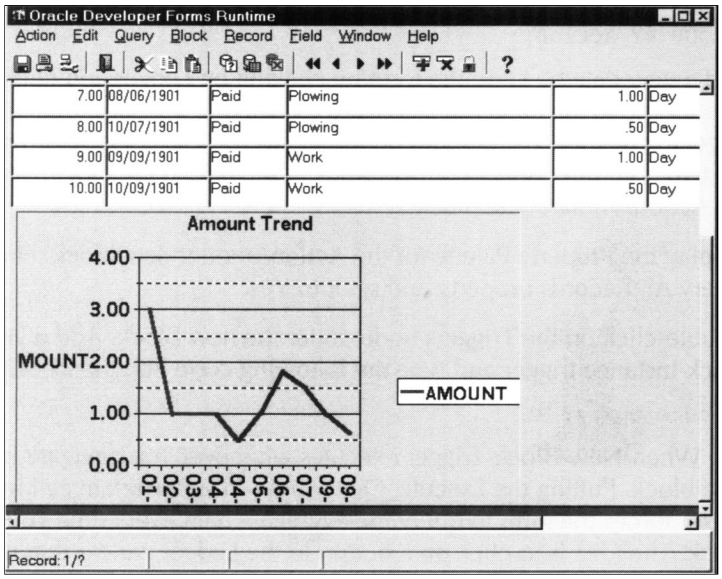


图4-58 带图形显示的Ledger表单

虽然有一些方法根据嵌套查询来创建数据块，但得到这种数据的最容易的方法是用适当的汇总聚集方式在数据库内创建视图，利用视图并使用 Data Block Wizard来创建这个数据块。它还把汇总聚集的处理移到服务器，这可以改善性能并减少网络信息通讯量。

注意 这个数据块和它的对应画布和窗口的创建要求使用一些高级技术，有关细节将在第6章学习。当前，只要完全跟随这个教程说明学习。

12) 用以下的在SQLPLUS或另外的SQL执行工具中作为Talbot用户执行的SQL在数据库中创建一个视图：

```
CREATE OR REPLACE VIEW ActionAmount (Action, Amount) AS
SELECT Action, SUM(Amount)
  FROM Ledger
 GROUP BY Action;
```

13) 在Ledger上单击，然后在 Add工具上单击，加一个新数据块到表单中。利用 Data Block Wizard找出ActionAmount视图(为了在浏览器看这个视图，不要忘记单击这个视图的单选按钮)和这两列。选择这两列包含在这个块中，这就创建了 ActionAmount数据块。

14) 显示ActionAmount数据块的Property Palette，找出Query All Record属性并把它设置为YES。

15) 在这个新块下，在Triggers结点上双击加一个When-New-Block-Instance触发器并把以下代码键入PL/SQL Editor：

```
Execute_Query;
```

每当导航到这个数据库时， When-New-Block-Trigger触发器执行。把Execute_Query内部的子程序调用放入这个触发器，使得每次人进入这个块时，表单都去查询这些汇总的数据。在导航到ActionAmount数据块上的图形显示项的 Ledger画布按钮上单击，这个导航就启动这

China-pub.com 下载

个触发器。因为这个数据块的Query All Records属性设置为YES，所以这个触发器查询所有的汇总数据，然后这些记录填充这个图形显示。

16) 通过在Object Navigator中选择Canvases结点创建一个新画布，并在Add工具上单击。命名这个画布为PIECHARTCANVAS.

17) 通过在Object Navigator上的画布对象上双击，显示这个空白的PIECHARTCANVAS。

18) 在PIECHARTCAVAS的Layout Editor的Chart项工具上单击，并在画布上拉出图形的长方形。Form Builder再次显示Chart Wizard。

19) 输入要显示的标题“ Amount by Action ”，选择带普通子类型的Pie(饼形)图，单击Next。

20) 选择ActionAmount数据块作为数据源并单击Next。

21) 选择Action项作为X轴或种类数据源，并单击Next。

22) 选择Amount项作为Y轴或值数据源，并单击Next。

23) 把这个图形显示作为ActionSummary.OGD文件保存，并单击Finish。

24) 重新命名Object Navigator的图表项为“ AMOUNTBYACTION ”，并把它移到CHART数据块。

创建ActionAmount数据块的最后一个步骤是加一个OK按钮到这个块。它有两个用途：第一，这个按钮显示在图形窗口中，让你解除这个窗口。如果没有这个按钮，就不能解除这个窗口。第二，这个按钮作为画布中的单个可导航的项将保存这个显示。如果没有一个可以将焦点转移到其上的可导航项，就不能显示带有它的窗口。

25) 在显示PIECHARTCANVAS画布的Layout Editor的Button工具上单击，并在你刚加的图表项下画一个小按钮。

26) 双击这个按钮，显示Property Palette，找出Label属性并设置它为“ OK ”。

27) 在Object Navigator中找出这个按钮，重新命名这个按钮为“ OKBUTTON ”。移动这个按钮使它成为ActionAmount数据块的第一项。双击Trigger结点，加一个触发器到这个项并加一个When-Button-Pressed触发器，把以下代码输入到PL/SQL Editor。

```
Go_Block('LEDGER');
Hide_Window ('CHARTWINDOW');
```

当用户按OK按钮时，Forms Runtime启动When-Button-Pressed触发器。Go_BLOCK内部子程序从当前块 (ActionAmount)走出进入Ledger块。这就从图形窗口除去了焦点。然后Hide_Window内部子程序从表单中除去这个窗口，所以不能再看见它。这个作用是关掉窗口

注意 与调用内部的Close_Window相比(它并不在Oracle Developer中)，似乎这个方法有点“华丽”，但所有这些步骤都是有意义的。第6章详细讲述导航过程和如何支持焦点从数据块移到数据块，从窗口移到窗口。这种与窗口同步作事情的方法，以及画布用精确方法在下面显示数据块的方法使得Form Runtime总有单个焦点。乍看起来它似乎是混乱的，但是一旦懂得它的用法，就成为Form Builder编程的第二个特性。

接下来必须为画布创建一个新窗口。

28) 在Object Navigator中找到Windows结点，并在Add工具上单击，加一个新窗口，通过在缺省的名字上单击，把这个窗口命名为CHARTWINDOW。

29) 在窗口对象上双击，显示窗口的Property Patette，找出Title属性，设置为“ Summary

China-pub.com
下载

of Amount by Action ”。

30) 找出Primary Canvas属性，并从下拉列表中选择PIECHARTCANVAS。

这把画布和窗口联系起来，还需要确定画布的 Window属性设置为CHARTWINDO。

31) 在对象上右击，显示画布的 Property Palette，或从弹出菜单中选择 Property Palette。找出Window属性，并把它设置为CHARTWINDOW。

现在必须创建一个显示图形的按钮。

32) 显示Ledger画布，在Button工具上单击，并拉出画布上的长方形按钮，在这个按钮上双击，显示它的 Property Palette。把Label这个属性设置为“ Summarize ”，进入Object Navigator并把这个按钮移到CHART数据块(见图4-59)。

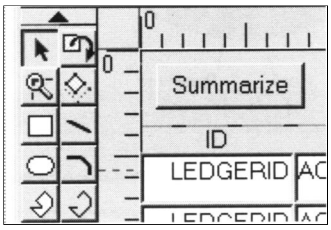


图4-59 设置Label属性为Summarize

33) 在Object Navigator中找到这个按钮项，找到这个项下的Triggers结点并双击它产生一个新触发器。添加一个 When-Button-Pressed触发器，把以下代码输入到PL/SQL Editor：

```
Go_Item('ACTIONAMOUNT.OKBUTTON');
```

Go_Item内部子程序导航到一个特殊的项，即是刚创建的 OK按钮，就把焦点移到OK按钮上，同时也移到它的画布上。这个画布也显示饼形图。小心使用单引号，而且不要拼写错了。应该用数据块的名字给这个项的名字加前缀，使得如果在多个数据块中有相同的项名也不会产生岐义。

当用户运行这个表单时，在 Summarize按钮上单击， Form Runtime启动When-Button-Pressed触发器，触发器然后把光标引航到 OK按钮，使饼形图画布显示在它的缺省窗口中。在Ledger窗口上弹出这个窗口，并显示饼形图。

当编译和运行这个应用程序时，在这个按钮上单击就会看到饼形图出现在一个单独的窗口上，如图4-60所示。

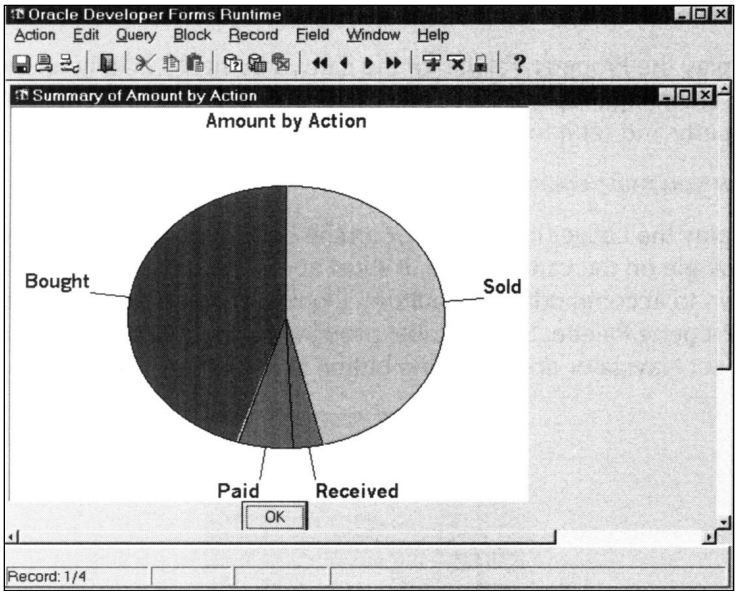


图4-60 Ledger表单饼形图

China-pub.com 下载

提示 可以通过选择File | Exit菜单项关闭这个窗口。你可能还希望使这个窗口作为对话框并加OK按钮(请参看第6章)来关闭它，还要在窗口的标题栏上提供标准的窗口关闭按钮。

4.3.2 在报表中创建图形显示

下面的教程教给你以下的技能：

- 如何在报表中创建图形显示。
- 如何使用Report Builder Chart Wizard。
- 如何利用报表的汇总信息产生图形显示。
- 如何在报表的首部确定图形显示的位置。
- 如何把显示作为单独的可再使用的模块来保存。

本教程创建一个饼形图，显示 Ledger事务在人员中的分布情况。饼形图可以使你通过人员在事务上的表现将人员分类。这个饼形图将出现在报表的首部 —— 在报表的重复记录前面的页面上。

1) 打开Ledger Summary报表，通过双击 Object Navigator的Layout Model结点，进入Layout Editor，在Header Section工具上单击。

2) 在Chart Wizard工具上单击，欢迎屏幕后的下一屏让你给你的图形确定一个标题，并选择要创建图形的类型(折线、饼形、条形等等)，如图4-61所示。

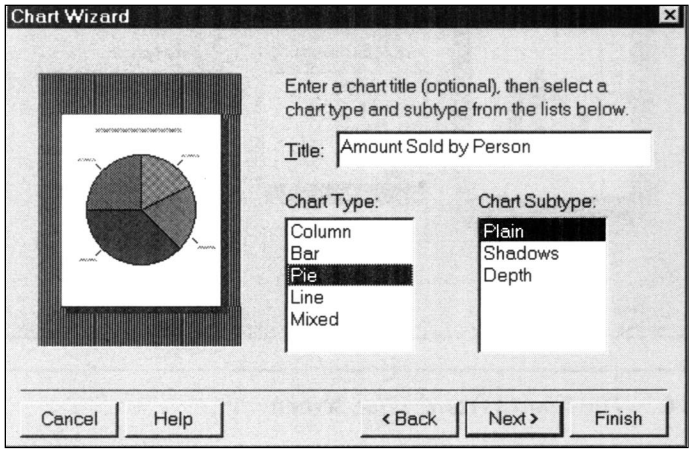


图4-61 Ledger Summary报表的Chart Wizard图表类型屏幕

3) 在标题域输入标题“ Amount Sold by Person ”。要为Ledger表单创建一个饼形图来显示按人员的数量分配，在第一列的 Pie上单击，在第二列的Plain单击，再单击Next。

所看到的下一屏显示报表域。当创建一个报表图形时， Chart Wizard假设你要显示报表数据模型中的数据。图4-62中的种类域屏幕显示出可以在图形显示中使用的报表域的列表。

4) 选择Name域来标识饼的扇形区(分类轴)，在Next上单击，进入值域屏幕，如图 4-63所示。

5) 选择SumAmountPerName作为Y轴 —— 图形用它来计算饼图扇区大小的数量，你需要每个人的Ledger项的和。在Next上单击。

China-pub.com
下载

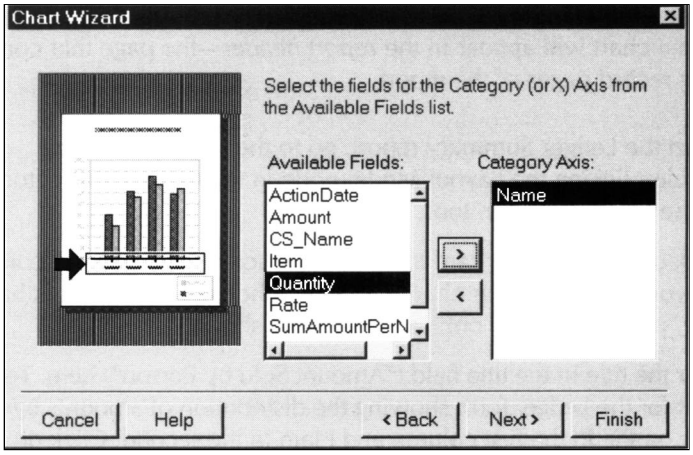


图4-62　Chart Wizard种类域屏幕

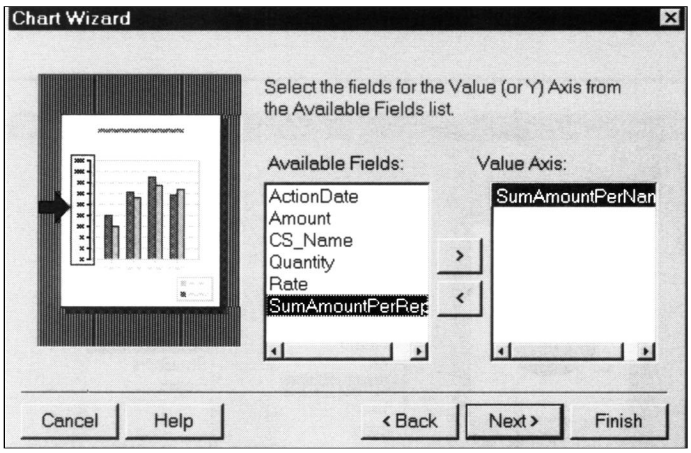


图4-63　Chart Wizard值屏幕

下一屏是打印频率，如图 4-64所示。这一屏让你在几个选项中选择报表图形的位置。在这种情况下，只要显示一次这个图形。

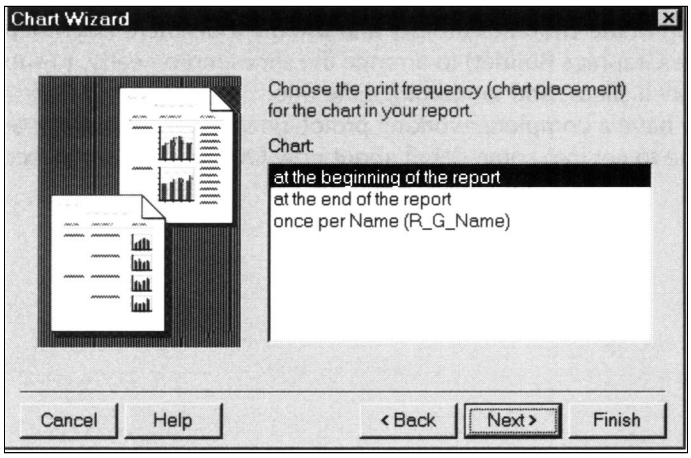


图4-64　Chart Wizard打印频率屏幕

China-pub.com 下载

6) 选择“ at the beginning of report ”，把这个图形放在报表的首部，在 Next上单击。

7) 把这个显示模块作为 AmountByPerson.OGD保存在工程目录中。在 Finish上单击，完成这个图形。

8) 运行这个报表，将看到图形出现在这个报表的第一页，如图 4-65所示。

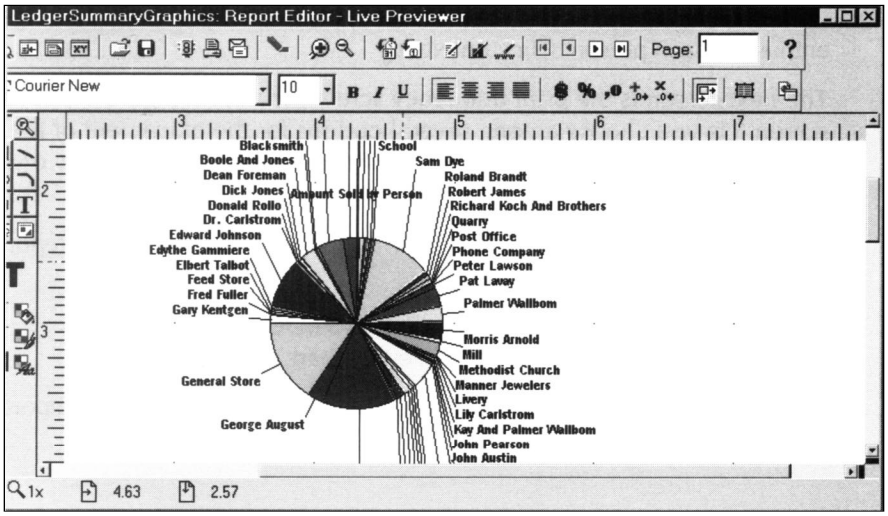


图4-65 Ledger Summary报表中按Person计算的销售量(Amonnt Solal)

这个特别饼形图离最优结构还差得远。要精心制作它，可以在 Graphics Builder中打开它，用那里的工具(有关Graphics Builder的详细内容，请参看第 8章)把这些扇区排列得比较精致，压缩许多非常小的扇区，并重新布置标题。

现在完成了表单、报表和图形的工作原形。现在可以进入有关 Oracle Developer如何完成所有功能的细节部分了。

China-pub.com
下载

第5章　高级进程、触发器和跨平台设计

第4章讲述了用Oracle Developer做的许多工作，本章研究这个产品的内部工作情况，展示一些特殊的进程，以及它们是如何工作而产生所想要的效果的。有了进程的知识，在某些细节上，就可以了解如何使用 Oracle Developer的事件处理特点，为以后的几章中的高级教程作准备。

注意　本章的辅导和例子给你说明如何以及在哪里的触发器上使用PL/SQL编程，将你自己的逻辑加到Oracle Developer的标准进程中。通常，这些标准进程就足够了。第4章讲述了如何完全不用编码建立一个工作应用程序。第11章和第12章详细地讲述PL/SQL编程。所以本章只是简单地提供示例的代码，而不去进一步充分讲述编程的细节。这些示例说明如何完成触发器的基本任务，你可以用在这里和在第 11章第12章学到的东西为基础，去建立非常复杂的应用程序。

在Unified Modeling Language(UML)中，活动的框图表示进程。活动框图通过进程表示工作的流动，很像旧的流程图。但是活动框图直接涉及对象的状态和它们之间的控制流。它们适合于说明Oracle Developer运行时的表单、报表和图形的进程工作流。在本章的第一张图中给出了一些例子，图中的文本足够评细地对活动图进行解释，使你能了解它。本章讲述在Oracle Developer中跨平台开发应用程序的一些有关问题。展示了要得到最大的可移植性如何设计表单、报表和图形。

5.1　Oracle Developer进程和触发器：它们是如何工作的

这一部分综述Oracle Developer是如何施用它的魔法的。前面几章介绍了 Oracle Developer数据库应用程序和对象的基本概念，本章讲述这些对象如何相互作用，象一个系统那样工作。

在设计Oracle Developer应用程序中，最实际的低级设计是决定将代码放在哪里。因为你不是根据Scratch来开发应用程序逻辑，所以必须使增加的东西适合于 Oracle Developer的逻辑。为了做到这一点，需要知道 Developer在哪运行你的代码，并在执行代码之前和之后发生什么。这一部分通过说明这些进程是什么，在这些进程内的什么地方激活触发器，清楚地告诉你在Oracle Developer进程的什么地方可以进行干预。

Oracle Developer应用程序是事件驱动的。当程序运行时，你和运行系统相互作用引发事件，事件又启动各种表单的进程。这些表单进程包含有预先编程的表单、报表和图形的缺省特性。在各种内部事件发生的过程中，运行时的系统通过运行你用 PL/SQL写的代码来处理它们：激活触发器。

你可以在高层次上继承设计师在这个产品中建立的一组假设和特性，而不必从头开始分析和设计。用一组预先编程的特性开始，这是一个很高的开发起点。本章的进程描述了 Oracle Developer应用程序的缺省运行时的特性。了解这些进程帮助你像 Oracle Developer那样思考，而不是根据Scratch来开发整个系统特性。如果你利用 Oracle Developer工作，而不是与它对着干，那么，大概会成功。

China-pub.com 下载

5.1.1 表单进程和触发器

在Oracle Developer的三个部分中，Forms有最成熟的逻辑。这部分概述Forms中不同的进程和触发器。

首先，必须了解一个基本的概念：导航和范围。

1. 导航

用户界面允许系统的用户工作。虽然这似乎是明显的，但其重要性在于了解表单开发结构的基本原理。Oracle Developer的表单是事件驱动的，这就意味着用户通过用户界面来控制进程，而不是应用程序来控制它。

在任何给定时间，Oracle Developer Forms的用户界面允许用户和单个对象相互作用。比如按钮、下拉列表(菜单)或文本域。单个对象是用户界面的一部分，是用户的焦点 (focus)。引用焦点的另外一种方法是引用游标的位置 —— 一种虚拟的指针，它指到有这个焦点的对象。光标是虚构的野兽，有时允许人看见它。例如，当游标出现在文本域时，它通常作为对平台输入文本的指示器，比如Windows中的细直条或Macintosh中的一束射线。其他项，比如图片或单选按钮并不明显地显示光标，所以使用“焦点”这个术语可以避免混淆。特别是术语“游标”至少指三种不同的东西，它可以指鼠标，SQL的游标(cursor)，或独立于表单的键盘游标。

注意 “单个对象”指的是单个应用程序对象。从表单进程的角度看，用户界面，比如窗口和菜单是不相干的，只有画布上的项目可以放置焦点。然而从用户界面的角度看，“焦点”像对对象一样，也适用于包含应用程序焦点的窗口。所以，如果焦点在文本域，那么，它也在包含文本域的画布的窗口上出现，而且也在应用程序的 MDI窗口出现。在这本书中，术语“焦点”指的是应用程序对象焦点，不是窗口的焦点。你可以从McGraw-Hill Web站点，www.Osborne . com和这本书的例子一起下载这些进程的综合参考材料。

用户通过从对象到对象移动焦点与应用程序交互，这个移动就是 navigation(导航)。通过从项到项、从记录到记录、从块到块，或从表单到表单导航，用户控制应用程序和任务。导航事件(进入到不同的项，进入到不同的记录，进入到不同的块，进入到不同的表单)构成了表单应用程序的基本事件。后面部分的许多进程或者包含这些事件或者是这些事件的结果。

在Oracle Developer Forms中导航的对象按导航单元的层次排列。表单包括一系列的块，每个块包含一组记录和一系列项。Oracle Developer Forms的导航活动是由于用户的活动而自动发生的导航。如果焦点在一个块的一个项上，而在这个块的另一个项上单击，那么则从项导航到项。如果项不再同一个块上，那么必须从数据块导航到数据块。如果每个数据块有不同的记录，那么你还必须从记录导航到记录。每一个导航事件都有不同的结果。比如某些触发器启动(在这种情况下，分别是When-Validate-Item、When-Validate-Block和When-Validate-Record(当确认项时，当确认块时，当确认记录时)。

对象的层次 —— 项、记录、块、表单 —— 导致一组进程，关系到导航的层次，比如说Navigate to Form Level (导航到表单层)。在这些表单进程的名字中的“层”(Level)指的是在对象层次结构中的导航单元：

- Navigate to the Form Level(导航到表单层)
- Navigate to the Block Level (导航到块层)

China-pub.com
下载

■ Navigate to the Record Level (导航到记录层)

■ Navigate to the Item Level (导航到项层)

每一个表单进程都把焦点从当前单元移到指定的目标。这个进程识别包含目标和“ enters ”的导航单元的下一个比较小的子单元，把焦点设置到那个对象上。

导航进程自动地确认当前的导航单元。例如从某一项导航到表单层，那么根据导致这个导航的特定事件来确认这个项。某些事件导航不确认，大多数导航有确认。大多数取决于单元——层，在这层上通过表单的 Validation Unit(确认单元)属性设置需要确认的表单。本章后面的“确认”部分，讲述基本的确认进程。

虽然大多数表单导航是用户行为的结果，但进程也可以引起导航发生。例如在许多 Oracle Developer Forms触发器中，可以调用内部的子程序，比如 Go_Item、Go_Block.或Go_Record。在本章的后面，“导航的处理”比较详细地讲述导航。

注意 导航是Oracle Developer的关键概念。实际上，所作的每一件事情都和导航有关。虽然这个概念和进程其本身是相当简单的，但它们普遍存在，与许多有意思的而有时是不可预测的方法相结合，超过任何别的东西。在 Developer中导航是必须在字面上和图形表现上均要“照此办理”的，试图“对付”Developer的导航结构，总将发生问题。

2. 触发器作用域

可以按照Object Navigator的分层结构在项目层、块层或表单层定义 Oracle Developer Forms的触发器。触发器作用域是一组激活触发器的对象，它由拥有触发器的对象和属于这个对象的对象组成。例如，如果把一个触发器连接到一个块，那么这个块的所有项的事件都启动这个触发器。

注意 留心的读者可能发现，最后一个段落并没有提到记录层作用域。虽然有记录导航触发器，但并不是在记录上定义这些触发器，记录只在运行时存在，在Form Builder中它不能作为对象得到。因此，不能把触发器连接到“记录”上。因为数据块可以看作是记录的聚集，所以经常将面向记录的触发器连接到数据块上。这些块触发器由当前记录——有焦点的记录——启动，因此记录触发器没有实际的作用域。

如果在一个特定的作用域内有多个触发器有相同的名字， Oracle Developer按缺省值启动连接到层次结构中最低的那个对象的触发器。例如，如果把一个 When-New-Item触发器放在一个项和这个项的块上，那么 Oracle Developer启动连接在项上的触发器，而不理会连接到块上的那个触发器。可以通过改变触发器属性表中的 Execution Style(执行方式)属性改变特定触发器的属性。缺省值为 Override。还可以选择Before或After，Before规定触发器在任何较高级的触发器之前启动，而 After则是在较高层的触发器之后启动。次高层的触发器可以选择和当前的触发器一起启动。可以在较高层的作用域层上设置这些属性，使得在这些层上的所有的触发器可以在项目触发器之前或之后启动。

提示 利用Execution Style(执行方式)设置为Before或After使多个触发器启动，显著地增加了模块的复杂性。除非有十分充足的理由要这样做，否则应避免它，以使模块比较容易理解、调试和测试。

某些触发器只有在特定的层上定义它们才有意义。例如， When-Validate-Record并不适用于单独的项，只可以在块和表单层上定义它。 SmartTriggers特性识别那些很可能为特殊对象

China-pub.com 下载

定义的触发器。在Object Navigator或Layout Editor上右击想定义触发器的对象，引出弹出菜单包含一个SmartTriggers项，在这个项上单击，显示一个子菜单，列出对于这个对象最有意义的触发器。选择适当的项，显示这个触发器的 PL/SQL编辑器。子菜单还包含一个 Other项，显示标准的触发器选择对话框，可以列出所有的 Oracle Developer触发器(见图5-1)。

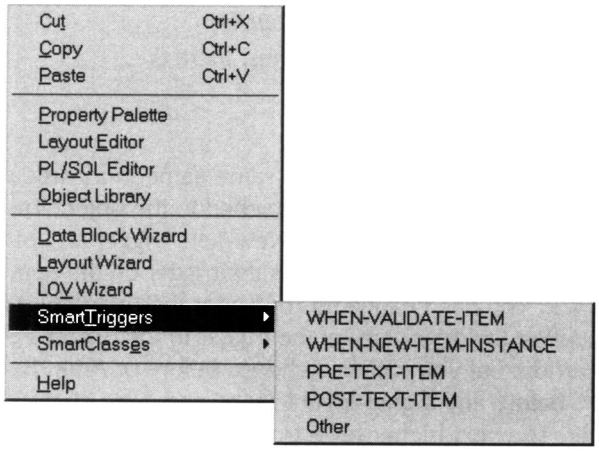


图5-1 Oracle触发器

也可以通过属性类继承触发器，而且可以引用已经定义过的触发器。这些触发器有一个相同的规则：仅有一个触发器执行。如果用项上的触发器取代 Property-Class触发器，则项触发器执行。对象组仅能够定义表单层的触发器，块和项的触发器和它们的块和项一起产生。

注意 为了简单起见，本章的图表不展示触发器和它们的判定。但是每一个触发器都会成功或失败，如果触发器成功，进程如流程图所示的那样继续，如果触发器失败，进程停止，带着错误返回。

现在已经对导航建立了一个概念，而且基本了解了触发器是如何工作的，如何利用SmartTriggers，以及如何修改触发器的缺省属性。下面该了解进程和触发器在实际的项目中都做了些什么。

3. 事务处理

表单事务是一连串的事件、进程和触发器，最终导致将数据提交给数据库或回滚变化。这一部分讲述事务中实际工作的基础进程。下面包括一些真实的进程。

1) 登记数据库　登记数据库的意思是通过一系列的 INSERT、UPDATE和DELECT语句把表单中正在进行的各种变化写进数据库。 Oracle Developer产生这些语句作为它的缺省处理的一部分。登记并不提交这些变化，但是它通常伴随提交或回滚作为缺省性的一部分。登记进程的作用是启动所有有效的触发器，所以每一次对数据库的登记都使存贮的数据生效。不管是否提交。

在缺省情况下，在数据库的一次登记后面是结束事务的提交或回滚。但是，可以利用POST内部程序来登记变化而不用提交。不用提交的登记的主要用法是，当从另外的表单打开一个表单时，可能在第一个表单中有未提交的数据，而第二个表单的存在可能正是为了加更多的数据到一个正在进行的事务中。

例如，可能要在一个会计事务中输入一项，这要求通过一个单独的表单在一个辅助表上

China-pub.com
下载

加一些详细的信息。在打开第二个表单时， Oracle Developer要求未完成的数据都要记入数据库以保证数据的完整性。因此，应调用项目的 Post in the when-Validate-Item触发器代码。

注意 第10章详细讨论多表单处理以及管理它的不同技术。

在详细讨论登记过程之前，对活动流程图作一简略的说明。图形中长方形是行为说明，说明内部的行为和至少一个转移。本章的活动图全部用内部的 Oracle Developer Forms进程作为行为说明，比如Savepoint 或Clean up cursors。箭头表示转移 —— 从一个行为说明转移到另一个行为说明或判断上。有两种特殊的转移，起源于黑圆圈的转移是一个空转移，表示图中活动的开始。在外带一个圆圈的黑圆圈处终止的转移是一个退出转移，表示活动结束。退出转移是“成功”还是“失败”取决于活动的结果。菱形表示判断，根据条件分支转移 (如果条件为真，流向一个方向，如果条件为假，流向另一个方向)，在判断处的一个转移旁边的文本是隔离(guard)条件。

图5-2是记入进程的简化活动图。

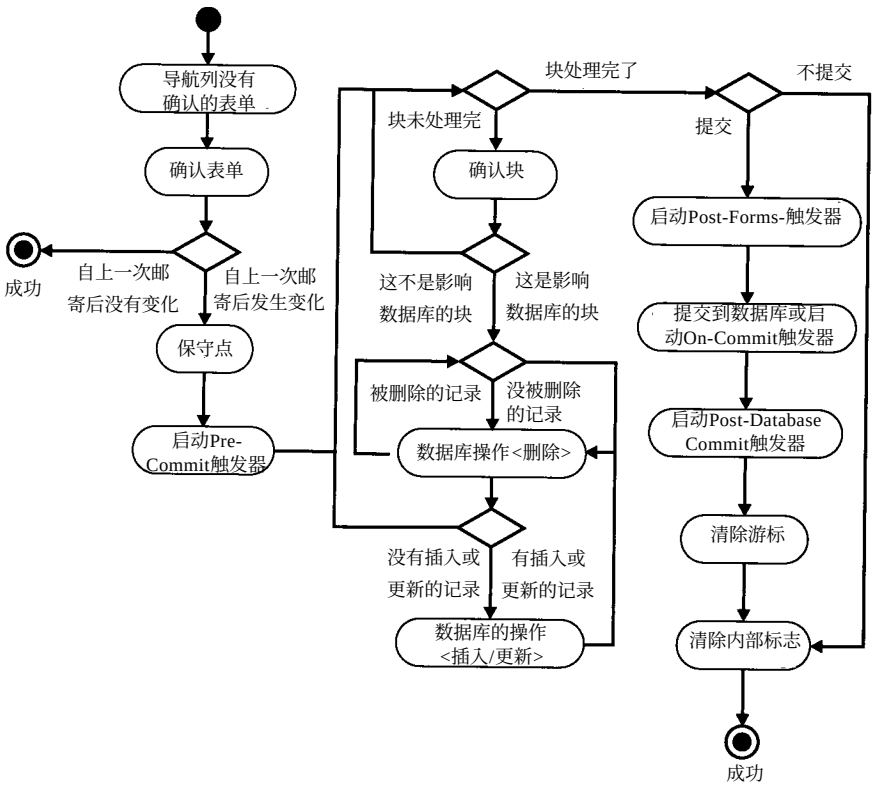


图5-2 记入过程的简化UML活动图

现在，详细描述记入过程。首先， Oracle Developer导航到表单层，然后确认表单，如果没有要求记入的变化，那么过程就停在这里。

然后Oracle Developer在记入之前发出一个保守点，保守点是 Oracle的下一个特性，它标识事务的一个点，如果需要你可以回滚到这个点，而不必回滚整个事务，这就允许 Oracle Developer记入多次而不用回滚所有的记入 —— 只要失败的那个邮件。

China-pub.com 下载

注意 除了Oracle，其他的数据库程序没有SAVEPOINT语句。所以如果要求数据库可移置，那么就不应该使用POST程序。而且在利用Open_Form或Call_Form内部函数调用另外一个表单时，Oracle使用SAVEPOINT。所以如果在调用的表单中提交失败，它只回滚到这个保存点。

在发出这个保存点以后，Oracle Developer启动Pre-Commit(预先提交)触发器，然后通过数据块循环。确认每一块，然后确定这个块是否有影响数据库的任何变化。如果这个块是个控制块，就没有这样的变化，如果这个块的基础表是一个不可更新的视图（带有一个连接或其他不合格条件的视图），那么只有在它上面定义了事务触发器，Oracle Developer才处理这个块。如果这个块没有变化的记录，Oracle Developer跳过这个块。

块处理将进程分两部分：通过在数据块记录中的循环的方式，处理被删除的记录行和处理记录的插入和更新。图 5-3展示了数据库操作过程的细节。数据库由 DELETE、INSERT、UPDATE语句和Pre-以及Post-operation触发器组成。如果需要控制数据库操作的话（例如，使用不同于Oracle的、有不同语法的数据库管理程序），可以用On-<Operation>触发器替代缺省的操作。

处理完所有的块以后，如果进程是提交，而不只是登记，那么 Oracle Developer启动Post-Form-Commit触发器，提交这些变化（或者启动On-Commit触发器)并启动Post-Database-Commit触发器。然后像所要求的那样，清除 SQL游标，然后Oracle Developer清除内部标志，包括把所有插入的记录标记为数据库记录，把任何变化的项标记为 Valid(有效)。

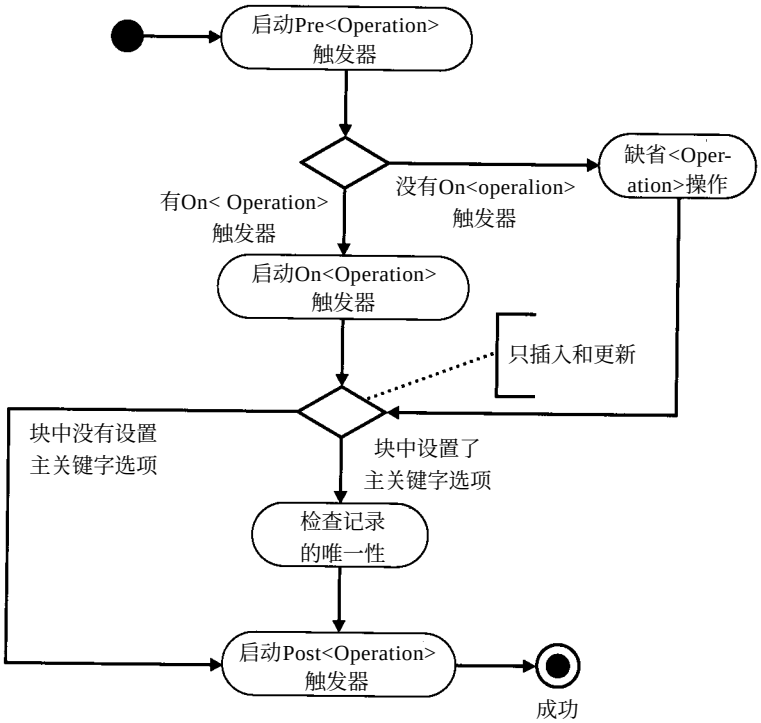


图5-3 数据库操作的UML活动图

注意 如果使用任何一个Pre或Post-Operation触发器来改变表单中与数据库有关的数据，

China-pub.com
下载

Oracle Developer可能不确认它们就把这些变化提交给数据库，因为它立即把每一样东西都标记为Valid(有效的)。当触发器启动时，重新确认可能会导致无限的确认循环，造成新变化，并要求更多的确认。

登记中最普通的触发器是Pre-Insert。Pre-Update、Pre-Delete、Post-Insert、Post-Update和Post-Delete(插入之前、更新之前、删除之前、插入之后、更新之后和删除之后)触发器，这些触发器包含有你根据要登记到数据库中的变化而写的代码，这些代码用于检查多表条件或操作其他表的数据，也可能包含有加到应用程序以跟踪特定表活动的检查更新。

注意 Pre-和Post-触发器的许多行为可能更适合于在服务器触发器里编码。就是说，可能想为操作前后的行为在数据库中加触发器。当这些触发器必须引用数据库内不存在的数据时，比如控制块的项值，或还没有移到服务器数据库的其他数据块的值，应该把这些数据移到应用程序的触发器。即如果需要在PL/SQL代码中引用一个块和项，那么可能需要在应用程序中编写触发器代码。使用这些触发器的另外一个理由是把信息处理从数据库服务器移到应用程序，使得到更好更有用的信息成为可能。

例如，Ledger表单让Talbot农场的员工在每天的日记中输入事务，在删除一行之前，Pre-Delete触发器包含有代码查寻用户关于删除事务的安全访问等级。如果用户没有合适的安全等级，则触发器失败，系统显示一个错误信息，告诉用户 (他或她)没有删除事务的特权。

```
DECLARE
  v_Level NUMBER = 0; -- lowest security level as default
BEGIN
  SELECT SecurityLevel
    INTO v_Level
    FROM User_Security_Level
   WHERE User = :Control.Username AND Object = 'Transaction';
  IF v_Privilege < 5 THEN -- delete only for security level 5 and over
    Message('You must have level 5 security to delete transactions.');
    Form_Trigger_Failure;
  END IF;
END;
```

2) 锁定 Oracle数据库管理程序通过锁定资源(比如锁定表和行)，允许同时访问数据而确保其完整性。锁定是Oracle实现事务的方式或工作逻辑单元。某些数据库应用程序锁定整个表，另外一些则锁定数据的物理页。Oracle Developer使用专门的行锁定，即在特定的行上锁定，这种锁定防止其他事务获得这一行，防止它们更新或删除这一行。在 Oracle 的SQL SELECT语句中加FOR UPDATE OF子句，就需要获得行锁定的信号。

按照缺省设置，在以下任一条件下，Oracle Developer获得行锁定：

- 当操作员改变数据库行中的基础表项的值时，包括从 NULL(空)值变化到non-NULL(非空)值。
- 当操作员通过Record | LOCK (记录|锁定) 菜单项为当前记录要求明确的行锁定时。
- 当触发器通过UPDATE(更新)或DELETE语句对数据库进行修改时。
- 当触发器调用 DELETE_RECORD、ENTER_QUERY(FOR_UPDATE)、EXECUTE_QUERY(FOR_UPDATE)，或LOCK_RECORD内部函数时。
- 当触发器包含显示的 SQLLOCKTABLE语句时。通常，这将严重影响应用程序事务的性能，如果没有绝对充足的理由，没有经过了解该事务的人对设计的专门检查，不应

China-pub.com 下载

该使用它。

如果Oracle Developer不能立刻获得锁定，那么它将在固定时间内反复试图获得锁定，然后询问是否前进。如果这种情况经常发生，那么请数据库管理员监控数据库弄清楚为什么得不到这个资源。通常，这是使用了明显的独占表锁定的结果，应当尽可能避免这种情况。

和任何事务一样，当事务提交或回滚时，数据库管理程序释放所有的锁定。因为 Oracle Developer使用保存点，在回滚后可能保留一些锁定，允许在进行必要的变化之后，只重做回滚的部分。

注意 如果在触发器明确地锁定查询行，那么在提交或回滚以后，需要在这些行上重新建立锁定来查询这些行。清除一个块或记录并不释放这些记录行所对应的行锁定。

如果在某一层定义一个On-LOCK触发器，那么每当Oracle Developer试图在这个层上获得锁定时，它就启动这个触发器。可以利用这个触发器来替换缺少的锁定机构 (如果有足够的理由这样做的话)。

警告 在对Oracle数据库运行应用程序时，避免使用On-LOCK触发器，因为可能由于替换了Oracle的锁定策略而降低事务的吞吐量。

通过块上Locking Mode(锁定方式)属性来控制锁定。在创建块时，这个属性的缺省设置是Immediate(直接的)，这意味着如果改变记录的基本表项， Oracle Developer将自动锁定当前的记录。可以把它改变为Delayed，意味着Oracle Developer等待到提交前才获得锁定。在某些情况下，当在查看一个值时，允许其他用户修改它，因为这可以改善大容量应用程序的事务性能。这种行为的风险是，在用户提交之前所做的任何工作被废除，会引起错误，被要求作一些意想不到的附加工作。如果它经常发生的话，可能引起混乱和令人气愤。在因特网上使用应用程序时，最好是利用Oracle Developer服务器的锁定和事务管理能力。使用这些特性的应用程序开发者不需要担心在他们的应用程序中管理锁定的困难和产生事务管理问题。

另外一种属性是在文本项中锁定记录。通常，对于那些不是基础表项的文本，就是说，那些和基础的数据库表的项不对应的文本项，在使用这个属性时，改变它们并不实行直接锁定。因为并没有真的改变数据库数据行的任何项。如果要改变项的文本，那么将这个文本项的Lock Record属性设置为Yes，即通知Oracle Developer锁定当前的记录。

4. 确认

“确认”是一个过程，是确定一个对象是否满足在定义表单时加在它上面的所有约束。在需要时， Oracle Developer自动确认项目、记录、块和表单。可以在触发器上附加另外的约束来作为确认触发器过程的一部分，或作为事务处理的一部分 (参看前面的部分)。当离开对象或当按下ENTER键或调用内部输入程序时，进行确认。当用户提交时或登记代码改变时登记过程自动确认。

(1) 确认单元和状态

可以确认项目、记录、数据块或表单对象，按照缺省设置，确认项将自动确认，这意味着每当离开项目时，即产生确认。可以设置确认单元为记录、数据块或表单，可以设置在它们的某一层自动确认。如果是这样， Oracle Developer在所规定的层上产生确认动作，而在此之前不做任何确认。例如，如果选择的是Form，则当运行表单的确认并且登记变化时才确认。通常，为了得到最好的结果，应该让确认单元在项目这一层。可以通过表单对象的 Validation

China-pub.com
下载

Unit 属性改变确认单元。

提示 Oracle公司建议，只有当用不同于Oracle的数据库管理程序运行应用程序时才改变Validation Unit属性。

在研究动态特性的细节之前，需要了解在 Oracle Developer Forms应用程序中与触发器有关的项和记录的不同状态。对象状态是在对象生命周期中某个点上的对象属性的设置。大多数Oracle Developer对象都有很多不同的属性，比如屏幕位置、宽度、高度等。与触发器有关的对象状态并不一定指的是这些属性，而只是指和触发器有关的特性。

下面的内容分别从构造单独的项一直到构造表单。每个对象均涉及其他的对象状态，通常按照层次向下构造。

(2) 项目的状态和确认

项目的状态必须由项目的值以及对这个值做了什么来决定。项目可以是一个新的、变化的或有效的。图5-4展示了确认过程的活动图

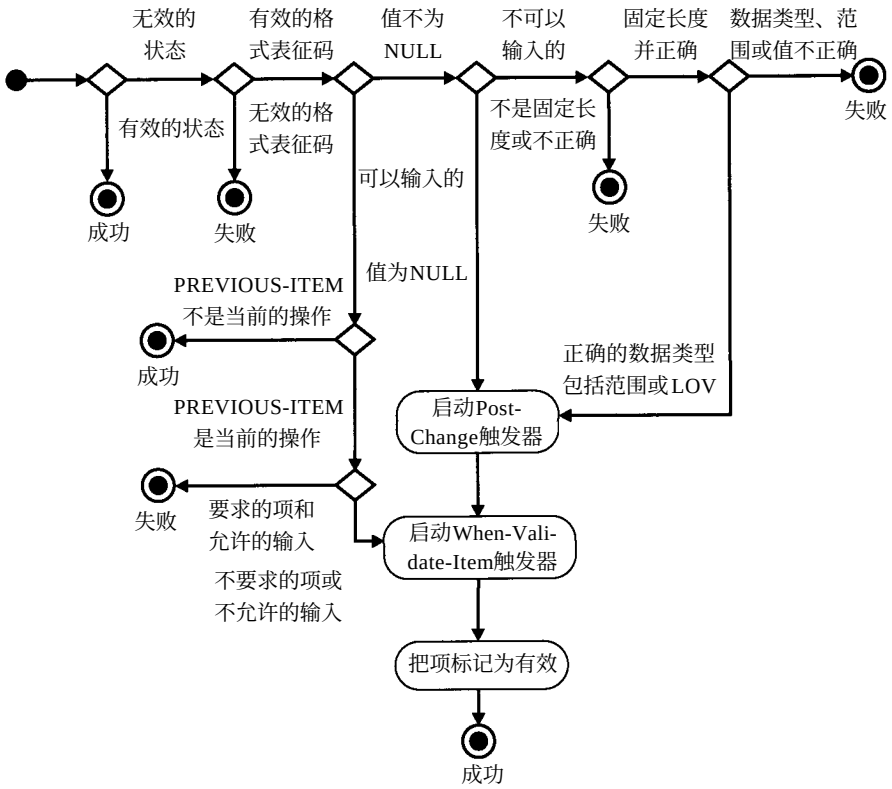


图5-4 项目的确认过程活动图

如在这个活动图中所看到的那样，确认主要是一系列关于各种有效性的判定。

可以用以下三种方法之一创建一个项目：

- Creating a record(创建记录)。创建记录中的项目作为New(新项)。Oracle Developer可以用缺省值填入或者从这个过程的其他项目拷贝它，但这个项目仍然是一个新的项目。
- Duplicating a record(复制记录)，把以前记录的值拷贝到一个新记录中，把新的记录设置为三个子状态中的任一个状态：新的、改变的或有效的。其状态是精确地复制以前

China-pub.com 下载

的那个记录项目的状态，即Oracle Developer所拷贝的那个记录的项目的状态。

- Fetching the record(取出记录)并立即使记录的项目有效，为每个项目执行 On-Fetch和Post-Change触发(后者是为了相容性而存在的，一般不应该使用它)。本章后面的“取出记录”将给出细节。

当用其中任何一种方法修改项目时， Oracle Developer把这个项目标记为Changed。项目的确认只对新的(New)和被改变的(Changed)项运行。当有一个New或Changed项目而且确认项过程的结果是True时，Oracle Developer启动Post-Changed和When-Validate-Item触发器，然后把这个项目标记为有效的 (Valid)。当确认项目的过程结果失败时，这个项目则保持原来的样子，而且游标继续停在这个项目上。

提示 不要使用Post-change触发器，因为Oracle要淘汰它，而应使用When-Validate-Item触发器。

项目由记录支配，所以清除个别项的唯一方法是清除记录。可以在项目的任何状态下进行清除。可以设置Clear-Item属性为项目的缺省值，但是，这并不能够从存贮器上把这个项目清除，只有除去记录才能把项目清除。

在确认一个项目时，该确认过程检查以下的属性，就像在图 5-4中看到的那样：

- Format mask:(格式掩码)：文本项目值的格式(可以是Valid(有效)或Invalid(无效))。
- Required and input allowed(要求和允许输入)：项值是否必须存在。
- Fixed lengh(固定长度)：项值是否必须是确定的那个长度。
- Data type(数据类型)：值的基本类型，比如数值型或日期型。
- Range(范围)：可能的最小值和最大值。
- List of values(值的列表)：值列表对象(LOV)的可能值的列表。
- Formula(公式)：如果域内有一个公式，那么这个公式是用于计算的。

如果确认了所有的内容，则这个进程为任何要附加确认的项目启动 Post-Change和When-Validate-Item触发器。如果触发器成功，就把这个项目标记为有效 (Valid)。

对于上面的项目确认属性的列表，可以看到在 When -Validate-Item触发器中只有很少的东西需要编码。假设你的数据库有 CHECK约束，它用简单的SQL表达式来确认在插入和更新中的个别项，所有这个内部检查都给定的话，会给 When-Validate-Item触发器留下什么？

一般地说，对复杂的多项确认，使用 When-Validate-Item触发器。通常，某种业务规则可能涉及多个项或者甚至涉及某种数据库查找。例如，在 Talbot农场的购买管理员为Buy(买)事务输入价格的操作中，会计部门对使用数据库价格的用户有一个限制： Authorized Purchase Limit(授权购买限制)，则可以在价格项目上对When-Validate-Item触发器编码，类似于这样：

```
DECLARE
  vPriceLimit NUMBER = 0;
BEGIN
  SELECT PriceLimit
    INTO :vPriceLimit
    FROM AuthorizedPurchaseLimit
   WHERE USER = AuthorizedUser;
  IF :Purchase.Price > vPriceLimit THEN
    Form_Trigger_Failure;
END;
```

When-Validate-Item触发器的另一个比较成熟的用法是用于为组合框的列表编程。组合框

China-pub.com
下载

是这样一个项目，它提供给用户一系列列表值，并且让用户利用 PL/SQL(这种技术超出了这本书的范围)输入不在列表中的值，在列表中添加新值时，可以使用 When-Validate-Item触发器。

(3) 记录的状态

记录有三种确认状态：

- New(新的)：在创建记录时，记录是新的。
- Changed(被改变的)：当改变了记录中的任何项时 (参看前面部分)，这个记录变成了被改变的。
- Valid (有效的)：在Oracle Developer确认了记录中的所有被改变的项或新的项并确认了这个记录后，或者在它从数据库中取出记录后，或者把记录提交到数据库以后，记录成为有效的。

复制记录即拷贝了源记录的状态。

记录的确认和登记只对被改变的记录进行。图 5-5显示了记录的确认过程活动图，有关登记过程的信息请参看早先的“登记数据库”的内容。

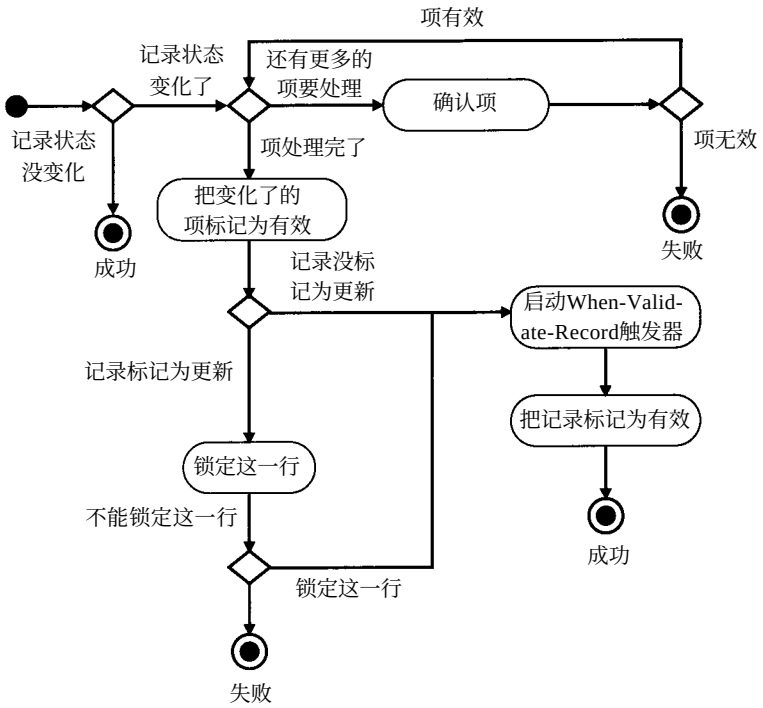


图5-5 记录的确认过程活动图

记录的确认进程要对记录中所有的项重复进行确认。如果其中的任何一项没能通过确认，那么用一个错误信息停止这个进程。在确认了所有的项以后，这个进程通过触发器把变化的项目标记为有效的，不必再重新确认它们。因此，如果记录被更新，那么进程锁定这一行 (有关细节请参看“锁定”)，然后启动 When-Validate-Record触发器，并标记这个记录为有效(Valid)。

When-Validate-Record触发器是一个逻辑位置，可以放置任何业务规则的检查，这些检查涉及一些不同的项目，这些项目在记录变成数据库的数据行时，其值必须一致。也可以利用

China-pub.com 下载

它来检查数据库或数据块其他一些项目的值。许多这样的约束和 When-Validate-Item触发器的情况一样，作为基础表上的 CKECK或FOREIGN关键字约束来表示，比作为 When-Validate-Record触发器的代码表示要好。另一方面，在某一个记录有某一方面的错误时，用户能立即得到反馈。触发器失败，显示信息会直接告诉用户在什么地方错了，而不是依靠关于约束失败的含义模糊的DBMS错误信息。

(4) 表单和块的确认

确认表单意味着确认表单上所有的块。在 Oracle Developer完成了块的确认以后，它检查在表单中是否带有触发器标记为 Change状态的任何项或记录，如果有，它把这些项或记录标记为Valid，则不用进一步确认它们。这个特性让你在确认触发器中做需要做的事情而不用无限循环。它可能对确认问题有影响，所以应该小心，使得这些触发器中的每件事情都正确 (见图5-6)。

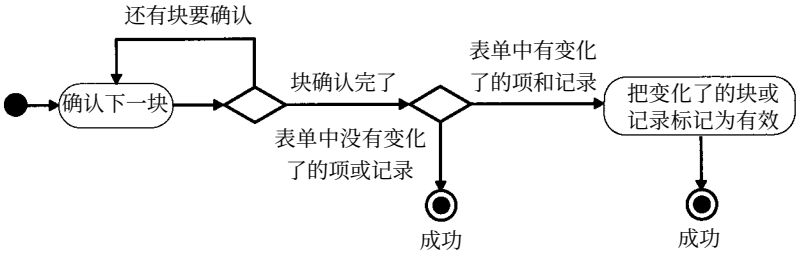


图5-6 表单和块的确认

确认块的过程随确认的单元而定稍有不同。确认单元可以是表单数据块或项。请参看前面“确认单元和状态”。如果单元是数据块或表单，那么 Oracle Developer确认数据块中的所有记录。如果确认单元是项或记录，它只确认当前的记录，如图 5-7所示。

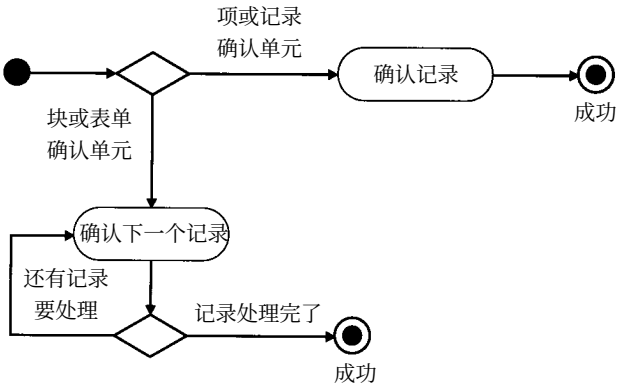


图5-7 确认当前的记录

5. 查询处理

查询是对数据库数据的请求。Oracle Developer Forms应用程序有内部的查询进程，一行代码也不用写，可以提供巨大的能力和适应性。下面这部分帮助你了解查询进程不同部分的逻辑。图5-8概括了表单的查询处理过程。

创建查询意味着指定想加到数据库上的逻辑条件。在 Oracle Developer中，如果要这样做(就是说，输入一个查询)，就应在一个块上创建一个实例记录，然后输入某些条件，将条件附

China-pub.com
下载

加到缺省的查询上。这种指定查询的方法叫做实例查询 (query by example)，因为它是用提供想看的数据的实例来规定查询。这个查询还包含一组条件和缺省的排序说明。如果希望的话，可以通过专门的对话框输入更复杂的 SQL条件，然后执行这个查询，并取出记录，在块中显示它们。这部分讨论三个主要的进程，每个进程有一些子进程，其他进程可以共享这些子进程。

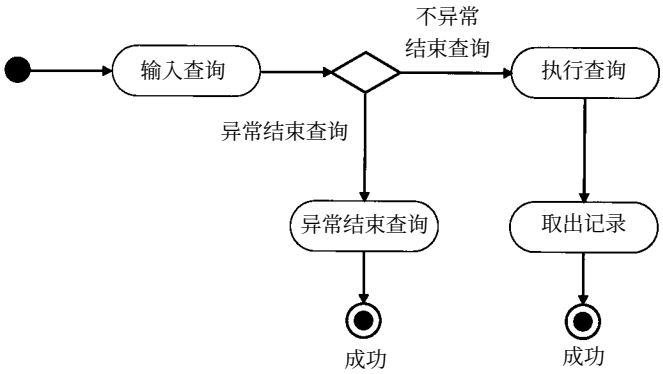


图5-8 表单的查询处理过程

还有两个进程：

- Count the query (查询计数)：这个进程为查询做准备，并要求服务器告诉你这个查询在指示信息中返回多少个记录。这个进程启动 On-Count触发器，有的话，替代计数进程。
- Abort and close the query (异常结束和关闭查询)：在输入查询时，你可能想不执行它就结束。这个进程启动On-Close触发器，如果有的话，它替代关闭进程。

1) 输入查询 对单个表查询的SQLSELECT语句有四个主要的子句：

```
  SELECT select-list
    FROM table
   WHERE query-condition
ORDER BY order-by-list
```

在Oracle Developer Forms应用程序中输入一个查询时，是通过实例建立这种类型的SELECT语句。 Selsct-list是一组对应于记录项的列项的名字。 Table是对应于块的基础表。order-by-list是select-list中的列项的列表，它规定被查询记录的次序，通过 ORDERBY子句块属性或通过Where对话框，可以交互式地规定 order-by-list。

Query-condition (查询条件) 是一组SQL表达式，规定要返回哪些记录，这组表达式有三部分：

- Default conditions (缺省条件)：用来规定作为块属性的缺省 Where子句的一组条件，可以通过触发器在运行时改变这些条件。
- Column conditions (列条件)：通过填写实例记录的列所输入的条件。
- Special conditions (特殊的条件)：通过Where对话框输入的条件。

可以通过数据块的Query Data Source Type属性，将查询建立在程序、事务触发器或显示的FROM子句上。在前两个例子中， Oracle Developer并没有使用各种条件，而是为程序提供自变量或引用触发器的项。在最后一个例子中，有所有的条件，但是，可以引用连接的任何列。有关使用程序作为数据源类型的更多的细节，参看第 6章中的“过程的封装”部分的内

China-pub.com 下载

容。

当用户输入查询时，他或她可以使用关系操作符 (>、<、=、！=等等)和其他的SQL特性作为列的条件的一部分。这允许一些相当复杂的查询不违反“通过例子”的方法。如果需要，用户可以避开单独的对话而直接输入 WHERE子句的SQL代码。

在输入查询时，Oracle Developer并不对项或记录进行确认。请使用标准的关键字定义而不是执行关键字触发器，并禁止各种功能键 (主要是记录导航功能和提交处理功能)。

注意 如果你希望能够覆盖一个特殊的关键字的话，可以把关键字触发器的“Fire in Enter-Query Mode”属性设置为True。但可能会引起相当大的混乱，所以，在这样做时，要以常规模式和Enter Query模式测试表单。

图5-9输入查询的进程流图：

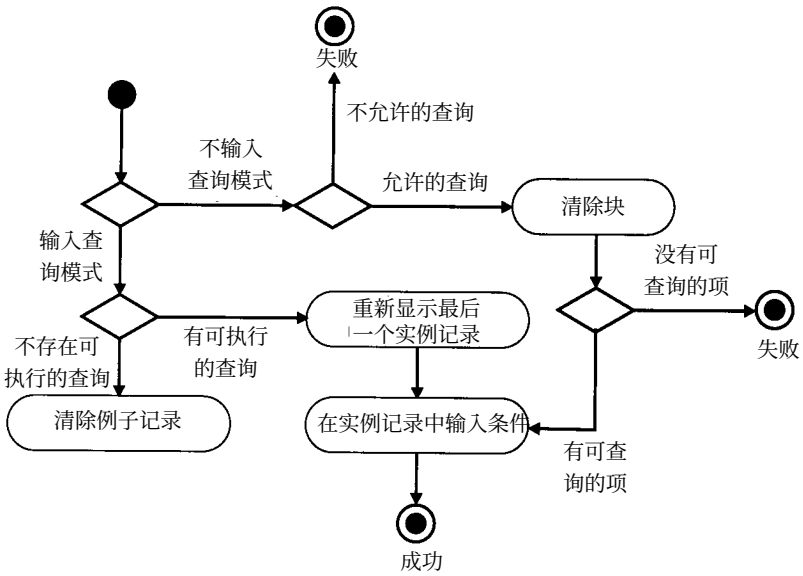


图5-9 输入查询的进程流图

Block属性表中的Query Allowed属性可以禁止对块的查询。可以通过设置 Database Data Block为NO规定一个块为control block(控制块) —— 没有基础表作基础的块。在块上至少应该有一项是来自数据库的基础表。

2) 执行查询 在输入查询条件以后，可以用菜单、工具条或为执行查询而创建的其他接口执行查询。当然，这并不是十分简单的。不首先输入查询也可以执行查询，这意味着你要利用缺省WHERE子句或查询程序，取回所有的行。可以当块中有变化的记录时执行查询，Oracle Developer会问你是否提交变化。下面讨论与取出的行的数目有关的一些复杂关系。

图5-10展示了查询的执行情况。为了简化，活动图没有说明把游标放在第一个记录的过程(或者当查询不返回记录时返回一个错误的过程)。

在执行查询时，Oracle Developer首先检查这个块是否允许查询 (像在前面部分讨论的那样)。如果允许，它导航到这个块，确认块中那些没有被确认的记录。如果块中有 Changed的记录，那么它提示询问是否提交或清除。

Oracle Developer然后触发这个块的 Pre-Query触发器。这个触发器提供另一个地方指定

China-pub.com
下载

WHERE列条件。Oracle Developer现在检查这个块是否有基础表，虽然重复了对这个块的检查，但你可以不做前面的检查而从另外的地方进入这一点，比如从计算查询的行进入。如果检查成功，则Oracle Developer建立SELECT语句并启动Pre-Select触发器。如果有On-Select触发器，就执行该触发器，否则执行 SELECT语句。启动Post-Select和When-Clear-Block触发器并刷新块中的实例记录，然后取回记录。

注意 如果因为某种原因你指定的查询导致错误和很长的处理时间，而你在 Windows Registry中设置了CNTL_BREAK=ON的话，那么可以用CTRL-C键终止这个查询。

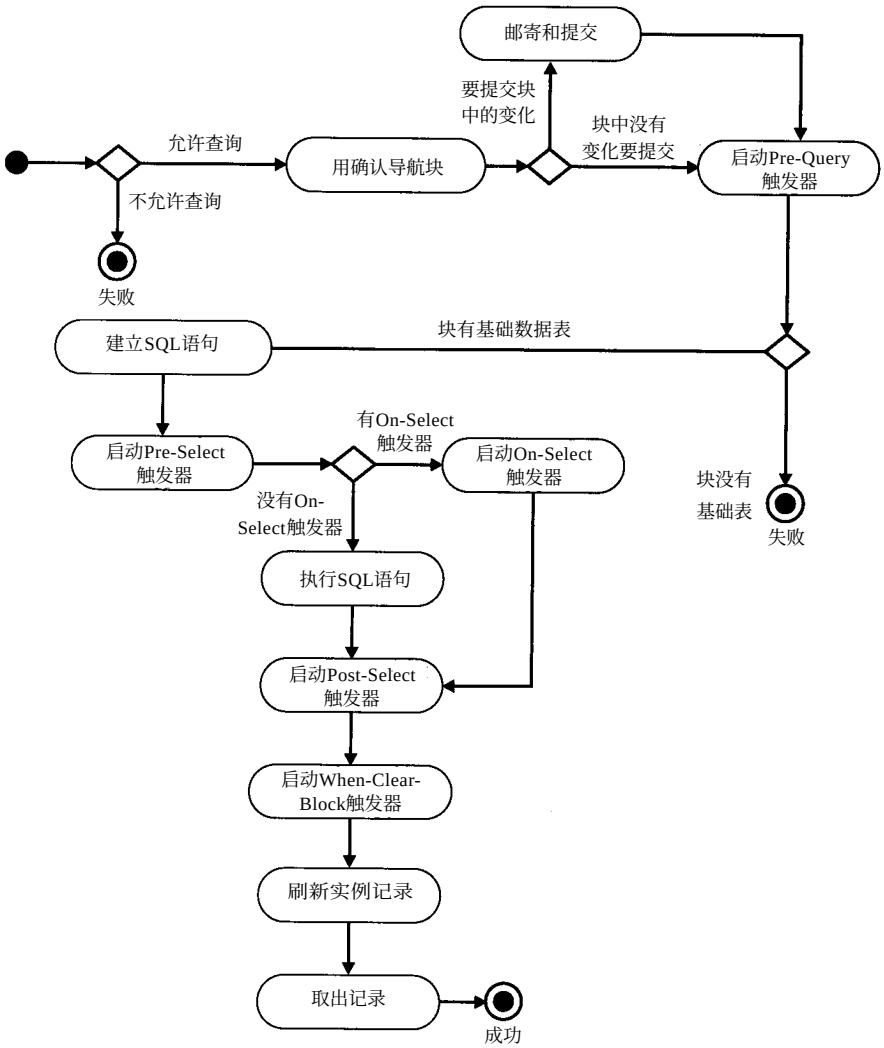


图5-10 展示执行查询进程的活动图

3) 提取记录 在执行SQL查询的过程完成以后，Oracle Developer就组织将记录提取出来。这是一个复杂的过程。它根据几个变量从数据库中提取数据行：

- 是否依赖数组处理。
- 查询从数据库中选择的数据行数。

China-pub.com 下载

■ 块中一次可见的数据行数。

■ 块在存贮器中存贮的行数。

■ 产生提取的操作(删除记录、滚动记录、或任何数量的其他可能的操作)。

如果有On-Fetch触发器，那么Oracle Developer启动这个触发器，而不从数据库中取一行放入记录缓冲器。

图5-11显示了提取数据的过程。首先检查已经取出放在缓冲器的等待行的列表。如果有这样的行，进程创建一个记录，并把列表中的当前行读入这个记录。如果没有等待的行，那么进程只是启动，或者从服务器取回更多的记录。如果当前的活动是 Clear Record (清除记录)，那就意味着需要取出另外一行填入这个空间。否则，要按照你在块的 Query Array Size属性里规定的数目来提取数据行。进程用适当的值设置块的 Record to Fetch属性。如果有On-Fetch触发器，则进程执行它，而且可以按照在块属性 Record to Fetch中规定的提取记录数去用 Get_Block_Propery访问那些记录。否则，进程提取出所要求的数目的记录并创建一个记录，把新的等待列表中的第一行读入这个记录。

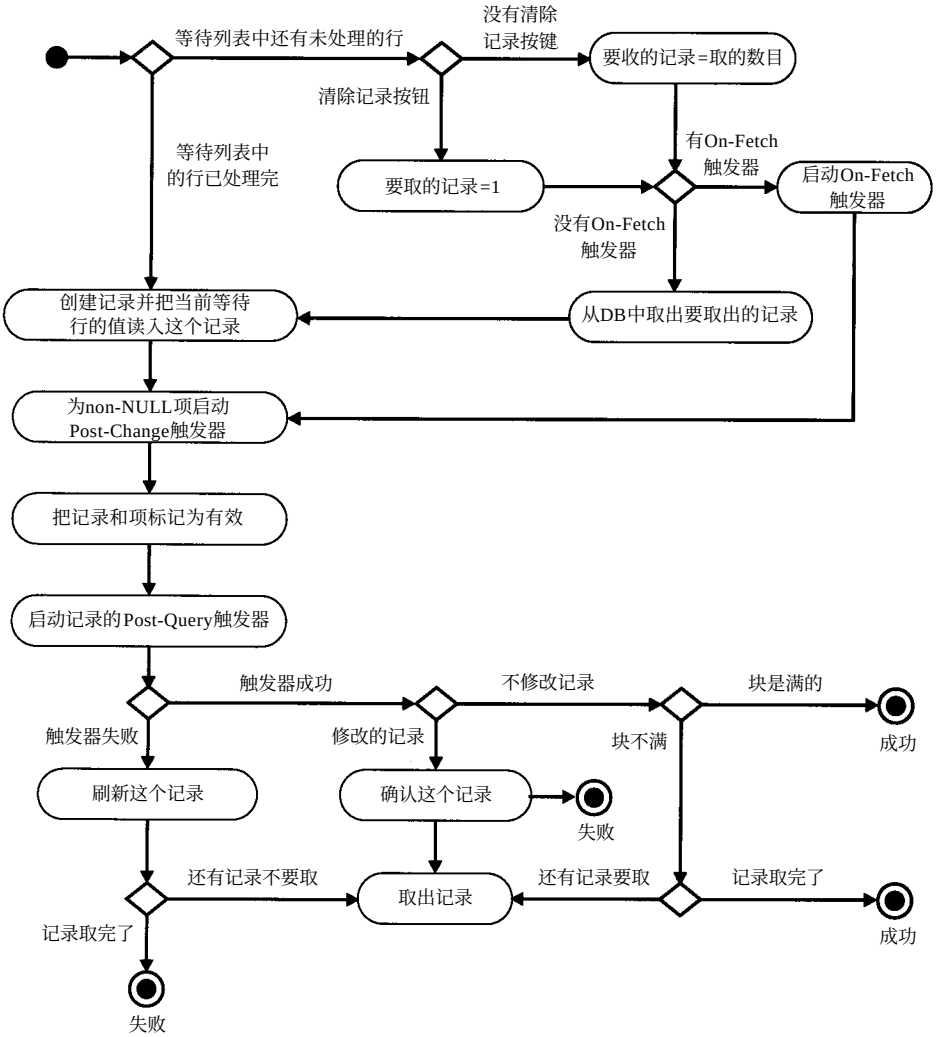


图5-11 提取记录的活动图

China-pub.com
下载

在记录创建完成之后，Oracle Developer循环通过新记录中的非空项，执行所有的Post-Change触发器。因为在这些触发器中有对处理的限制，而且它的存在只是为了向后兼容，所以应该避免使用它们，而只须依靠Post-Query触发器来完成取出记录的任务，比如确认或传送数据到记录的其他项中。在所有的项处理完后，进程把所有的项和记录标记为有效。

在把项标记为有效以后，进程为这个记录执行Post-Query触发器。例如，利用这个触发器去设置那些其值与基础表中的列的值不对应的项，或者用它确认项和记录的条件。如果Post-Query触发器失败，那么进程刷新这个记录，意味着它将不在显示的记录中出现。当发生这种情况时，进程重新启动自己，开始提取下一行。

注意 Developer并不通知用户它刷新了记录而且不显示它们，它只是“默默地”去做这件事，这可能会引起混淆。例如，如果把确认代码放入Post-Query触发器中，而数据库中50个记录中有10个记录在确认时失败，那么用户只能看到触发器成功的40个记录。这就意味着用户不能访问那些更新的记录。如果这正是你希望的，那很好。否则，除了在这个触发器确认失败时缺省调用Form_Trigger_Failure以外，还要做某些事情，例如，可能要显示错误警报或在表中记录这个错误。

Post-Query触发器的主要用法是用从数据库表中查找的值去填写非数据库项。例如，在人员表中，人员的地址信息由嵌套对象类型的若干列组成。你可能希望按照信封地址的约定在连续的和格式化的单个域内显示这个信息，而不是单独地显示所有这些列。如果街道地址只有一行的话，你也可能希望取消Street2(第二行街道地址)。你把个别的值取到项中，但是却不能个别地显示它们。取而代之是使用如下的代码，在Lodging块的Post-Query触发器中填充数据库的非数据库文本项，然后显示地址：

```
:Lodging.DisplayAddress := :Lodging.Street1 || CHR(10);
IF (:Person.Street2 IS NOT NULL) THEN - only show Street2 if it isn't null
  :Lodging.DisplayAddress := :Lodging.DisplayAddress || :Lodging.Street2 || CHR(10);
END IF;
:Lodging.DisplayAddress := :Lodging.DisplayAddress || :Lodging.City || ", " ||
  :Lodging.State || "  " || :Lodging.PostalCode;
```

提示 ASCII代码10是回车，它可以打断地址部分之间的行。CHR函数是一个Oracle函数，它返回ASCII字符的整数值。

如果Post-Query触发器成功，那么进程根据触发器是否修改了记录来决定做什么。如果记录被修改，则进程要对这个记录进行确认。如果确认失败，则用一个错误信息终止提取进程，否则，进程继续提取更多的记录。

如果块是满的，意味着Number of Records Buffered(缓冲的记录数目)属性设置的存贮器不能保存更多的记录，则当前的提取停止。如果块不是满的，而且要提取有更多的记录，那么进程从开始到提取下一个记录循环。

注意 实际上很少超过存贮器的限制，除非你把Number of Records Buffered(缓冲记录的数目)设置为(除了)缺省外的某个值。应该不干涉这个属性，除非愿意承担为了证实你的方法所要求的彻底的测试。

6. 导航的处理

当用户到处移动应用程序时，有几个触发器可以起到特殊的作用。触发器启动的进程包含有创建缺省值的各种方法，比如，在第一次输入一个块时，创建缺省的新记录。

China-pub.com 下载

进入对象，则启动该对象的触发器有：Pre-Form、Pre-Block、Pre-Record、Pre-Text-Item，离开对象时启动的触发器有：Post-From、Post-Block、Post-Record、 Post-Text-Item。

移动到不同的对象时启动的触发器有： When-New-<Obiect>-Instance、When-New-Form-Instance、When-New-Block-Instance、When-New-Record-Instance、When-New-Item-Instance。这些触发器在要求某种初始化的情况下，非常有用。例如在鼠标环境下对实现有条件的导航就非常有帮助，当表单用具体示例说明对象时，它们对定位到某处特别有用 —— 比如要定位到叠状画布的项或对话框的项。这是因为这些触发器允许使用带限制的内部子程序比如Go_Item和Go_Block，而导航触发器不允许这样。在本章后面有这方面的更详细的内容。

在启动一个新表单时， Oracle Developer启动When-New-Form-Instance触发器。具体地说，当导航到表单的第一个可导航的块的第一个可导航的项以后，那个触发器被启动。这就是意味着所有的导航触发器，包括 Post-Form、Post-Block、Pre-Form、Pre-Block、Pre-Text-Item等等，在When-New-Form-Instance触发器以前启动， When-New-Form-Instance触发器在把控制返回给用户之前启动，即正好让用户输入数据到表单以前启动。

顶层表单的When-New-Form-Instance触发器的共同的任务是控制 MDI窗口的大小。例如下面的When-New-Form-Instance触发器代码将窗口设置为 600×300个像素(如果Coordinate System是像素的话)：

```
Set_Window_Property(Forms_MDI_Window, WINDOW_SIZE, 600, 300);
```

当导航到不同块的项时， Oracle Developer启动When-New-Block-Instance触发器，启动它是在导航触发器之后和在把控制交给用户输入以前。可以利用 When-New-Block-Instance触发器来初始化内部的块元素，比如那些数据库不自动填充的列表和记录组。也可以在确认块以后，利用它控制导航，因为离开一个块通常意味着进入另一块。

当移动到不同的记录上时， Oracle Developer启动When-New-Record-Instance触发器。每当把焦点移到一个块中的记录并希望发生某件事情时，可以利用这个块的这个触发器。触发器名字中的“ New ”(“ 新 ”)在这里是“ 不同 ”的意思。每当移动到记录时而且不管移入它多少次， Oracle Developer均启动这个触发器。这个触发器在任何导航的触发器之后但在把控制返回给用户输入之前启动。可以利用 When-New-Record-Instance来初始化数据块记录的内部元素，或者可以在前面的记录被确认之后利用它导航。 When-New-Record-Instance的主要用法是在记录确认以后导航到新表单或另外一个块或项。因为 When-Validate-Record不允许使用限制的内部程序，比如Go-Record，所以必须依靠When-New-Record-Instance来控制导航。例如，可以在 When-Validate-Record触发器中设置一个控制项或全局项，然后在 When-New-Record-Instance触发器中检查它，导航并重新设置这个项。

在得到用户的输入以前，移到一个作为最后目标的不同项时， Oracle Developer启动When-New-Item-Instance触发器。当输入焦点移到某个项上时，如果想激活触发器，可以使用这个项上的这个触发器。在导航到达某个项并停在这个项得到用户的输入以前时，这个触发器并不启动。当移到报警域或菜单项时它也不启动。你将发现这个触发器对在导航完成以后调用受限制的内部程序特别有用，它对表单的特性有非常好的控制。

7. 注册处理

Oracle Developer Forms应用程序的注册和注销要在几个地方激活触发器。注册进入一个表单应用程序中，要设置各种选项、连接数据库、为当前表单的所有块设置列的安全性。图 5-12

China-pub.com
下载

是注册进程的活动图。进程检查是否已经注册，如果没有注册，则加以处理。在启动 Pre-Logon触发器以后，进程检查连接是否有效。如果没有连接，并且存在 On-Logon触发器的话，它就启动这个触发器。如果没有这种触发器，它将尝试连接到服务器上。在三次努力均不成功之后就失败。

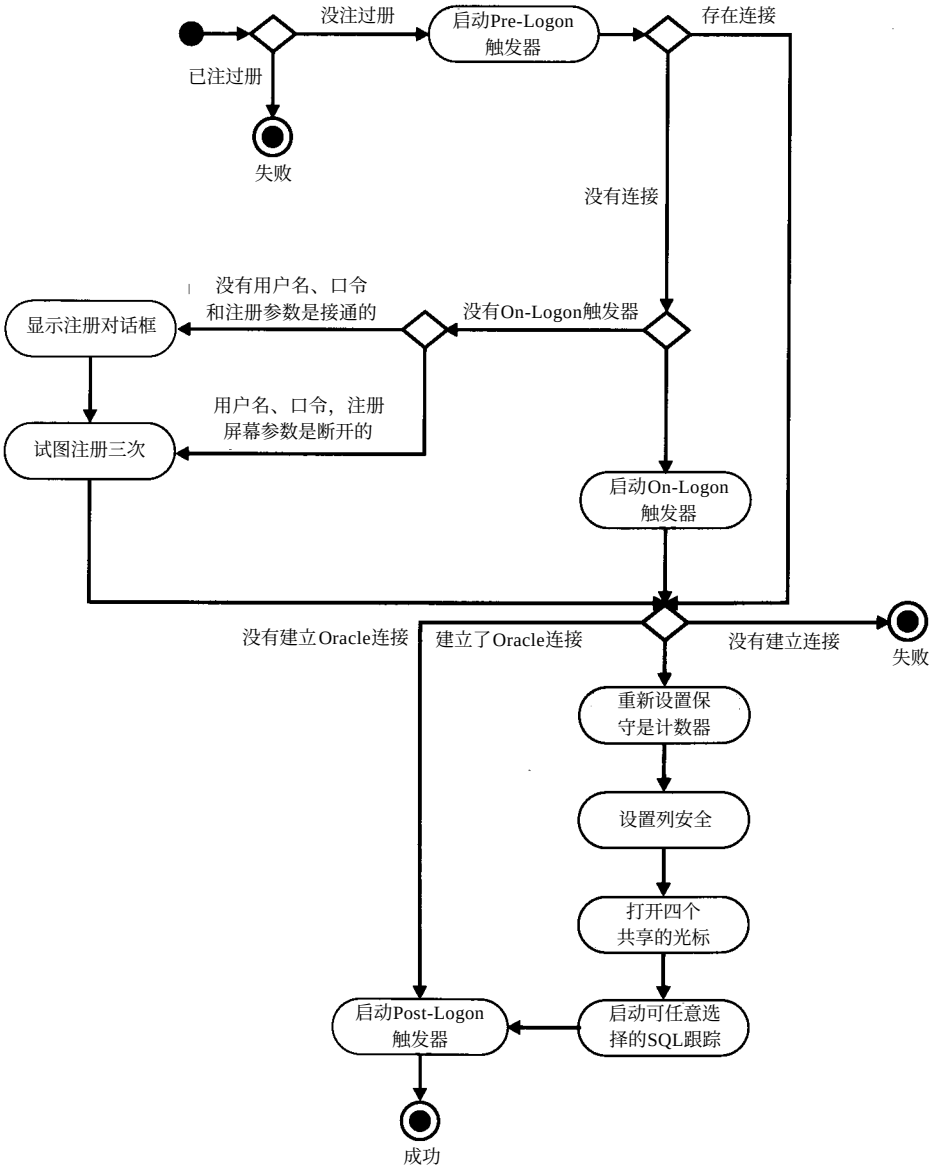


图5-12 注册活动图

如果要连接到Oracle数据库服务器上，和某些其他的数据库管理程序相反， Oracle Developer建立Oracle专用客户方的对象(保存点、列安全性、共享游标、 SQL跟踪)。Oracle Client Adapter (Oracle用户适配器) 使得第三方的数据库管理程序可以利用这些特性。有关详细信息请参看第14章的有关内容。

在重新设置保存点计数器之后，注册进程通过表单中所有的块，检查各个列的安全性约

China-pub.com 下载

束。在想要约束的块中，设置Columu Security(列安全)属性为True，使得可以对它进行安全性检查。在任何这样的块中，操作员必须在服务器的数据词典中有列层的更新特权，或者注册进程把这个项的Update Allowed属性设置为off。可以通过定义一个On-Colunu-Security触发器定义安全进程。

提示 Oracle公司建议不要使用这些属性，因为使用这些结果是使你的模块更加复杂。

实施了安全性检查以后，进程打开四个的游标，如果需要的话便启动跟踪，并启动 Post-Logon触发器。

注销一个表单应用程序比注册简单得多。即注销清理与数据库有关的存贮器并注销数据库。首先，进程启动Pre-Logout触发器，然后关闭Oracle游标。如果有On-Logout触发器，那么进程执行这个触发器，而不是注销服务器和启动Post-Logout触发器。

客户机/服务器应用程序的一个共同任务中把这个应用程序与不同的用户名相连接。当没有用用户名、口令或连接字符串启动Oracle Developer应用程序时，Oracle Developer显示初始表单并提示需要这些信息。

连接成功后，可能希望允许注销并连接到另外不同的用户名上，缺省菜单或表单中的有效活动集没有这部分功能。可以在工具条按钮项上用类似于以下的编码写一个象 When-Button-Pressed那样的触发器：

```
DECLARE
   vUser     varchar2(80);
   vPassword varchar2(80);
   vConnect  varchar2(80);
BEGIN
   Logout;
   Logon_Screen;
   vUser := Get_Application_Property(USERNAME);
   vPassword := Get_Application_Property(PASSWORD);
   vConnect := Get_Application_Property(CONNECT_STRING);
   IF vConnect IS NOT NULL THEN
      Logon(vUser, vPassword||'@'||vConnect);
   ELSE
      Logon(vUser, vPassword);
   END IF;
END;
```

当用户在这个按钮上单击时，这个触发器显示标准的Oracle Developer注册屏幕，在屏幕中输入新的用户名、口令和连接字串，并按该用户名为用户注册。如果想在输入的信息中进行错误检查的话，可以用自己设计的对话框替换标准的注册屏幕，

注意 在试图用另一个用户名注册之前，你必须先注销，否则会因为重复注册而失败。这意味着如果用户提供了一个错误的用户名、口令、或连接字串，应用程序将不连接到数据库。应该通过某种警告或信息，确保让用户了解这个情况，就象On-Error触发器那样。(有关细节请参看“信息处理”部分的内容)。

8. 数据块和记录的处理

数据块是表单中所发生的许多事件的中心。数据块是一个对象，它是一组项目的源。大多数数据块映射到服务器上的单个基础表上。但是也可以有控制块，这些块不映射到数据库表。利用控制块把不是来自数据库的项目加到表单中，形成不适合连接到数据库的块。在任

China-pub.com
下载

何情况下，数据块都是包括了项目的这种基本结构，而且它提供了它所包含的项目组的聚集层。可以利用这个层次完成许多不同的动作和作用。

除了前面讨论的广泛的查询手段以外，有一些基本的数据块进程启动应用程序的主要部分。

虽然创建记录很简单，但它是一个主要的进程。图 5-13展示了这个进程的活动过程。

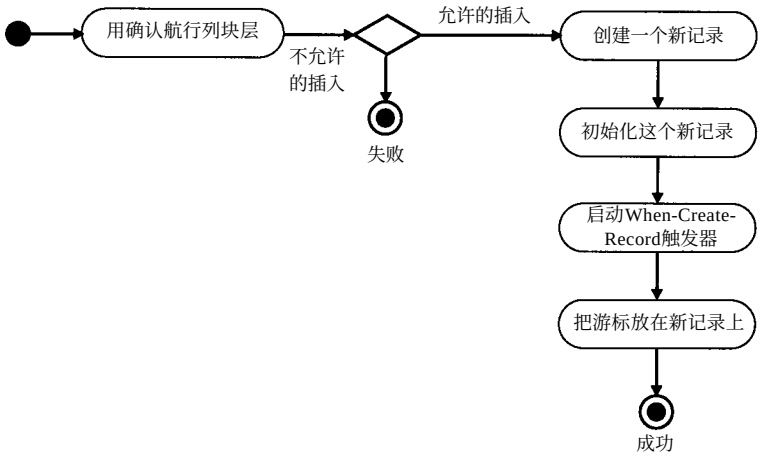


图5-13 创建记录活动图

在导航到带有确认的块的层次时，进程创建一个记录并初始化这个新记录（如果这个数据块允许创建新记录的话），然后进程启动When-Create-Record触发器，并把游标放在新记录上。可以用各种方式为改变了的项和记录做标记。有关的细节参看前面的“确认”部分的内容。对于Oracle Developer应用程序的事务处理，这个进程是重要的。这个进程的活动见图 5-14。

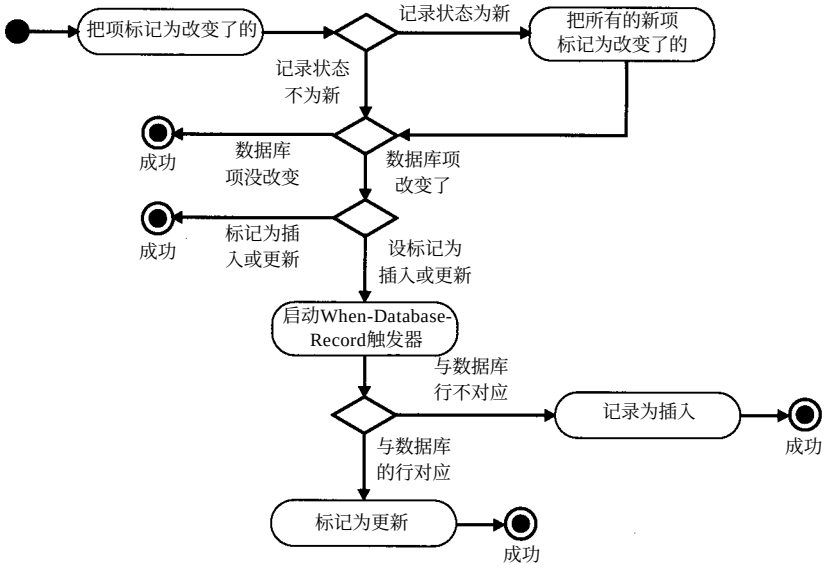


图5-14 进程活动图

进程首先将项目标记为“被改变的”。如果记录是“新的”，那么进程把这个记录的所有项都标记为“被改变的”。检查那些标记为“被改变的”数据库项，如果有这样的数据库项，

China-pub.com 下载

而且Oracle还没有把这个记录标记为插入或更新，那么进程启动When-Database-Record触发器，然后把数据库行(通过查询来自数据库的行)标记为Update(更新)而把其他的行标记为Insert (插入)。

清除数据块时先确认这个块，并提示是否要提交。如果希望数据块登记入库并且要提交，则启动When-Clear-Block触发器，刷新当前记录，并把游标放在当前块上。清除记录则有比较复杂的逻辑关系，要重新排列块和显示中保留的记录。在这个进程中启动 When-Remove-Record触发器是最重要的事件。

9. 接口和按键事件的处理

在Oracle Developer Forms应用程序中有几十个接口事件。庆幸的是，这些事件的有关逻辑和它们在应用程序中的处理不很复杂。通常，事情的发生是引发事件 (单击鼠标、按下按钮、检查复选框、关闭窗口等等)，以及由与这个事件有关的触发器触发引起。

利用这些触发器，可以为表单应用程序的大多数接口事件定义所发生的事情：

- 用户接口控制：可以为块中的不同项，比如，图像按钮、复选框、单选按钮和列表，定义所发生的事情。
- 鼠标事件：可以定义在鼠标单击、鼠标双击、鼠标下移(mouse-down)、鼠标上移(mouse-up)等操作后做些什么。
- 按键事件：可以定义各种功能键：Clear Block、Insert Record、Previous Block、Enter Query (清除块，插入记录，前一块，输入查询)等，定义键盘功能键(F0-F9)的功能。

注意 用Key-Others触发器去取代任何功能键时，功能键可能会有触发器，但在任何的表单层都没有定义这种触发器。可以利用这个触发器关闭所有键的功能，例如当用户处于对话框中时就可以这样作。

10. 主-从协调

Coordination-causing(协调原因)事件是主-从关系(master-detail pair)中的主块的某个事件，它使主块的不同记录成为当前记录。

Oracle Developer处理所有事件的实际过程是通过某些缺省产生的触发器来进行的，这些触发器是：On-Clear-Details、On-Populate-Details、On-Check-Delete-Master。Oracle Developer堆积这些触发器，并在下一个事件以前把它们当作堆放的一连串进程的一部分来处理。这个特性与表单应用程序其余的逻辑相比有点不一样。如果打算修改这些触发器，由自己来做协调工作，要准备进行逻辑实验，直到完全了解它为止。

当定义主-从关系时，表单创建On-Clear-Details触发器。当引起协调事件发生时，如果这个关系的Auto-Query属性是False，那么将自动清除从属块。

当定义主-从关系时，表单也创建 On-Populate-Details触发器。当引起协调事件发生时，它自动填充从属块。

当在关系属性表中把 Master Deletes属性设置为Non-Isolated时，表单创建 On-Check-Delete-Master触发器。这意味着如果有和主记录联系的从记录的话，要能够防止删除这个主记录。

还有一些逻辑必须设置Auto-Query，它允许延迟填充从属部分直到导航进入从属块为止。

提示 在重新安排表单数据块时，可能遇到这些触发器不一致的问题，可以利用 Data

China-pub.com
下载

Block Wizard通过删除和重新创建关系在任何点重新产生它们。也可以不用向导而通过它的属性表来创建关系对象并直接输入连接条件。如果在数据块模式框架外工作，则必须人工做这个工作，因为没有框架的块不能再使用Data Block Wizard。

11. 信息处理

在Oracle Developer Forms应用程序中有两个触发器影响信息的处理： On-Error和On-Message。前者是在Oracle Developer试图显示错误信息时启动，而后者是在Oracle Developer试图显示指示信息时启动。可以利用这些触发器来处理错误和信息。在触发器编码中，可以访问内部函数，从而返回各个部分的信息：

- Error_Code 或Message_Code ：返回错误编号。
- Error_Text或Message_Text：返回文本信息。
- Error_Type或Message_Type：返回信息的种类。
- DBMS_ERROR_Code：返回服务器错误编号，它可能和表单的错误编号不同。
- DBMS_Error_Text：返回服务器信息，它可能和表单的错误信息不同。

如果需要，可以抽取信息的任何元素并在进一步的处理中使用它，或者提供你自己的错误处理。通常，这样做的方法是在表单层对 On-Error触发器和On-Message触发器进行编码。这些触发器调用标准库的程序来格式化和显示信息。例如，下面是一个 On-Message触发器，它取代标准的表单信息处理：

```
IF Message_Type = 'FRM' THEN
  Handle_Forms_Message(Message_Code, Message_Text);
ELSE
  Handle_Database_Message(DBMS_Error_Code, DBMS_Error_Text);
END IF;
```

这个触发器对数据库信息和表单信息加以区别，并调用标准库中带有适当的信息代码和文本的子程序，这些子程序将使用标准的格式和专门的信息用于你的应用程序中。例如，如果在打开另一个表单以前不想显示来自登记变化的信息“ Database apply complete：<n>records applied OK ”，可以创建如下所示的Handle_Forms_Message程序：

```
PROCEDURE Handle_Forms_Message(pCode IN NUMBER, pText IN VARCHAR2) IS
BEGIN
  IF pCode = 40404 OR pCode = 40405 THEN
    null; -- do nothing for this message
  ELSE
    Message('Forms '||To_Char(pCode)||':  '||pText);  -- format message
    Message('');  -- stack message to display alert, clear status bar
    Synchronize;  -- display message
  END IF;
END Handle_Forms_Message;
```

这个程序忽略40404号和40405号信息，利用显示警报而不显示信息条上的信息的编程技巧，显示任何的其他信息。如果信息堆栈中有多个信息要显示，那么 Forms用第一条信息弹出一个警报，然后在状态条上显示最后的信息。这个程序利用这个特点使表单显示警报中的所有信息。有一些事件你可能想做或者可能不想做，这取决于应用程序的编程形式，内部同步机制告诉Forms采用信息堆栈同步显示，就是说，现在立即显示信息。

有三种Forms信息：工作信息，指示信息和错误信息。有关特殊信息的资料，请查询Forms联机帮助。可以用下面一行PL/SQL代码取消工作信息。

China-pub.com 下载

```
:SYSTEM.Suppress_Working := 'TRUE';
```

可以用ON-Message触发器处理指示性信息，在 Message_Code，Message_Text和Message_Type中有这些信息的资料。可以用 On-Error触发器处理错误信息。在 Error_Code、Error_Text和Error_Type中有关于错误的资料。显示信息的细微控制来自 System.Message_Level变量的使用，这个信息的行值表示Forms信息各种不同的严格程度，如表5-1所示。

表5-1 Forms信息的严格级别

严格程度	说 明	信息实例
0	任何信息	
5	明显的条件	在第一个记录上
10	程序的错误(要做的事情超出了命令的范围)	域是满的。不能插入字符
15	设计错误(要做的事情不能通过设计在表单里做)	试图更新一个不允许更新的域
20	触发器失败或阻止用户前进的条件	不能在视图中插入或更新数据
25	阻止表单正确执行的条件	由于以前的回滚，更新不能进行。清除一个记录
>25	Forms始终显示的错误	缺乏存贮器

这个变量的值确定了Forms显示什么信息。表单只显示所设置的以上等级的信息。例如，如果把变量设置为‘5’，那么Forms就禁止严重程度为0或为5的信息。利用这个变量，可以关闭那些被认为只会把用户弄糊涂的警告信息。

```
:System.Message_Level := '5';
```

每当要使用适当的触发器时，可以接通和关闭信息。可以查阅联机帮助，找出各种信息的严重程度。

警告 不要将一连串的操作的Message_Level设置为大于零。要尝试限制这个变化的范围，只需在可能发生混淆信息的短时间内消除混淆信息。如果长时间关闭信息，将发现调试和确定发生的问题非常困难，因为缺乏信息，你不可能知道真正的原因。

12. Restricted Built-in(受限制的内部子程序)

Oracle Developer的自动导航能力使得完成编程任务非常容易。然而另一方面， Developer复杂的导航和事件驱动处理具有潜在的无限循环。当 Oracle Developer Forms正在导航时(如果能够导航)，有可能陷入循环——你导航，导致表单导航，这又引起你导航，这样不断继续下去。庆幸的是， Oracle Developer认识到了这种可能性，它使用有限的内部子程序使得很难出现这种情况。乍一看，这个特性似乎可能使编程更困难，但是真实的情况正好相反。

Restricted Built-in是在导航时开始执行的内部子程序。 Oracle Developer规定在响应导航启动触发器时使用Restricted Built-in是非法的。在实际中，这种限制通常在触发器中引起运行时的错误，它使你要使劲地去考虑在这个 Oracle Developer的“限制”周围发生了什么并如何去工作。稍为思考一下，去熟悉这些限制，这样可能对避免这些问题有所帮助。

代替在导航触发器中调用被限制的内部子程序的是，把调用放在由导航触发器单独触发的触发器中。例如，不应在 Post-Query触发器中调用Go_Item来将游标移到不同的项，而应该把Go_Item调用放在这个块的Key-EntQry触发器内。如以下所示：

```
Enter_Query;
Go_Item ('Ledger.ActionDate');
```

这个触发器由用户输入查询时触发，而不是由缺省的查询输入动作触发。在用户执行这

China-pub.com
下载

个查询和Forms提取了所有的行之后，控制返回到这个触发器，这个触发器再把它转送到Ledge.ActionDate项。因为只利用按键也可以进行类似的动作，所以还需要在这个块上定义Key-ExeQry触发器。

```
Execute_Query;
IF :SYSTEM.MODE = 'NORMAL' THEN
Go_Item ('Ledger.ActionDate');
END IF;
```

如果你也想这么做，最好的开始位置是用一组 New-Instance触发器(When-New-Form-Instance、When-New-Block-Instance、When-New-Record-Instance和When-New-Item-Instance)。彻底了解这些触发器的用法，可以节省相当多的时间。

例如，比方说，有一个确认处理的问题，根据当前记录的状态要求导航到一项或另外一项，做这种确认处理的地方明显是这个项上的 When-Validate-Item触发器(Ledger.Action)。当确认发生时，必须把焦点移到另外一项，但是 When-Validate-Item触发器不让调用Go_Item！

为了处理这种情况，需要使用“全局变量”的编程形式。建立一个控制块 (数据块的Database Data Block属性设置为NO)和一个文本项，可以用来当作是被调用的 vDestination的变量，调整它的尺寸使得可以容纳任何项的名字。通过利用以下语句，把对 Go-Item的调用放在Ledger.Action的When-Validate-Item触发器中：

```
:Control.vDestination := 'ActionType.Action';
```

这个变量将包含对话框的画布上的项的名字 (在这种情况下是ActionType.Action)。在这个块中，或者在按照顺序排列的下一个项的 When-New-Item-Instance触发器中，有以下代码：

```
IF :Control.vDestination IS NOT NULL THEN
  Go_Item(:Control.vDestination);
  :Control.vDestination := '';
END IF;
```

当用户移动出Ledger.Action项时，When-Validate-Item触发器启动，设置vDestination变量。然后这个触发器成功地处理这种情况，完成导航并把焦点移到下一项。在那里，启动 When-New-Item-Instance触发器。这个触发器到达被命名的 ActionType.Action项，然后重新设置变量。这后一个步骤确保能够继续正确处理进一步的导航，否则，当移动到 Ledger块的下一项时，将只是弹回到对话框。

这种逻辑允许某些富有想象的策略。如果Oracle Developer能够“思考”的话，这里给出的例子就是对Oracle Developer“思考方法”的简单研究。不要企图使系统屈从你的意愿，应该了解它的进程是如何工作的并顺着它走。

提示 另外一种没有实际导航的导航方法：创建一个项作为指示标志，在项或表单层创建一个When-New-Item-Instance触发器，然后检查标志并导航到指定的项。当执行确认触发器时，设置这个标志，导航进行。当焦点移动到一个新项时， When-New-Item-Instance触发器启动，检查标志，并导航到指定的项。

警告 一个不做工作的“解决办法”是利用 When-Timer-Expired触发器。可以设置这个触发器在很短的时间内进行，使得它正好在导航停止后启动。这个触发器常出现在Oracle Developer的各个部分，因为在导航中触发器是不会触发的。计时器触发器在导航停止后几乎立即触发，然而，几乎还不够。任何熟悉多线程系统的人将告诉你，这

China-pub.com 下载

是一个很糟的想法。如果系统忙，不管时间间隔多么短，别的触发器可以先启动。如果是在 web 上使用，这更是个问题。因为每个计时器事件要求通过网络到达 Application Server。所以应该使用进程逻辑，而不要使用技巧去解决这个特别的编程问题。

表5-2、表5-3和表5-4分别列出了不能使用被限制的内部子程序的触发器，可以使用被限制的内部子程序的触发器和一组被限制的内部子程序。

表5-2 不能使用被限制的内部子程序的触发器

On-Check-Delete-Master	Post-Block	Pre-Insert
On-Check-Unique-Trigger	Post-Change	Pre-Logon
On-Close	Post-Database-Commit	Pre-Logout
On-Column-Security	Post-Delete	Pre-Query
On-Commit	Post-Form	Pre-Record
On-Count	Post-Forms-Commit	Pre-Select
On-Delete	Post-Insert	Pre-Text-Item
On-Error	Post-Logon	Pre-Update
On-Fetch	Post-Logout	When-Clear-Block
On-Insert	Post-Query	When-Create-Record
On-Lock	Post-Record	When-Custom-Item-Event
On-Logon	Post-Select	When-Database-Record
On-Logout	Post-Text-Item	When-Image-Activated
On-Message	Post-Update	When-Remove-Record
On-Savepoint	Pre-Block	When-Validate-Item
On-Select	Pre-Commit	When-Validate-Record
On-Sequence-Number	Pre-Delete	
On-Update	Pre-Form	

表5-3 可以使用被限制的内部子程序的触发器

Key-<Whatever>	When-List-Changed	When-New-Item-Instance
Key-Fn	When-Mouse-Click	When-New-Record-Instance
Key-Others	When-Mouse-DoubleClick	When-Radio-Changed
On-Clear-Details	When-Mouse-Down	When-Timer-Expired
On-Populate-Details	When-Mouse-Enter	When-Window-Activated
When-Button-Pressed	When-Mouse-Leave	When-Window-Closed
When-Checkbox-Changed	When-Mouse-Move	When-Window-Deactivated
When-Form-Navigate	When-Mouse-Up	When-Window-Resized
When-Image-Pressed	When-New-Block-Instance	
When-List-Activated	When-New-Form-Instance	

China-pub.com
下载

表5-4 受限制的内部子程序

Block_Menu	Delete_Region	Help	Post
Call-Input	Down	Insert_Record	Previous_Block
Clear-Block	Do_Key	Last_Record	Previous_Form
Clear-EOL	Duplicate_Item	List_Values	Previous_Item
Clear_Form	Duplicate_Record	Main_Menu	Previous_Menu
Clear_Item	Edit_Text_Item	Menu_Next_Field	Previous_Menu_Item
Clear_Message	Enter	New_Form	Previous_Record
Clear_Record	Enter_Query	Next_Block	Scroll_Down
Close_Form	Execute_Query	Next_Form	Scroll_Up
Commit_Form	Execute_Trigger	Next_Item	Select_All
Convert_Other_Value	Exit_Form	Next_Key	Select_Records
Copy_Region	First_Record	Next_Menu_Item	Show-LOV
Count_Query	Go_Block	Next_Record	Terminate
Create_Queried_Record	Go_Form	Next_Set	Up
Create_Record	Go_Item	Open_Form	Update_Record
Cut_Region	Go_Record	Paste_Region	

5.1.2 报表进程和触发器

Oracle Developer Reports进程比前面讨论的表单进程要简单得多。用报表做的大部分工作，特别是重复循环，就是用数据模型组、重复的模式、报表类型这三部分的组合来进行说明。有关报表的创建和编程的细节，请参看第 7章的内容。有一些特定的操作，特别是涉及条件判断的操作，必须在报表触发器中采用定制方法进行程序编制。图 5-15展示了报表进程，它包含有几个触发器的触发点。 Before和After触发器的主要用途是创建一个报表使用的处理表。例如，可以用SQL语句或PL/SQL代码创建一个临时表，然后用这个表的值作报表，这样作的一个例子是资金平衡表报表。资金平衡表报表建立一个平衡表，并从初始的平衡表和事务表中汇总会计信息。

报表进程使用给定的报表运行系统的命令行参数启动。如果用户提供了某些参数，则Oracle Developer启动输入的确认触发器，这个触发器是通过参数属性工作表检查和 /或改变报表实例的参数来触发的。运行时参数表单让用户为报表输入运行时选项。 Oracle Developer启动Before Form和After Form触发器，以及可能再次启动确认触发器。

设置参数以后， Oracle Developer编译这个报表并启动Before Report触发器。执行报表的逻辑比图5-15显示的要复杂。用户可以介入的地方是在和报表上的任何对象联系的 Format触发器上，以及在打印完每一页 (最后一页除外)以后启动的Between Page触发器上。

打印完最后一页后，报表提交结束这个事务，然后启动 After Report触发器。

可以在Previwer(预览器)上运行报表，也可以只把结果发送目的地。如果选择预演报表，则可以交互式访问报表。可以利用某些触发器来完成交互操作，特别是利用Action 触发器(在“作用触发器”部分有更多的内容)，可以通过这种方式把报表当作交互应用程序来处理。应该知道，在预演器中使用类似于‘向前引用’那样的操作可能会引起触发器以不可预料的方式启动。

China-pub.com 下载

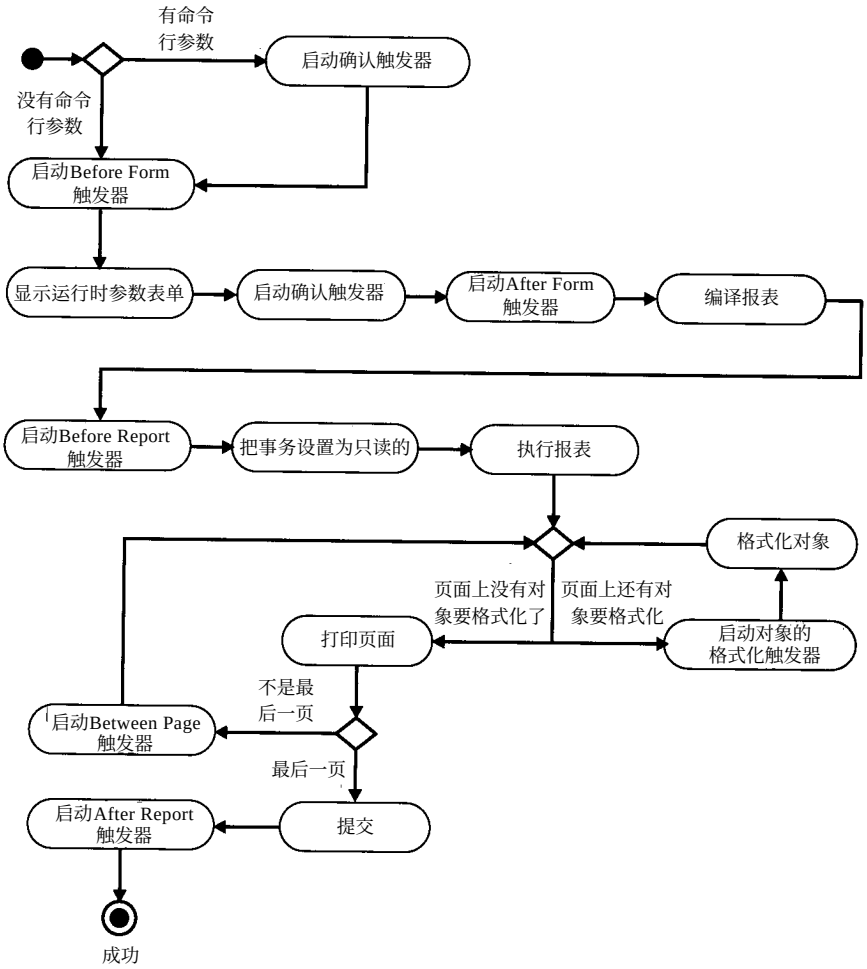


图5-15 报表进程的活动图

1. Report触发器

Report触发器(Before and After Report、Between Pages、Before and After Form)让用户在报表层和页面层上对报表处理进行程序上的访问。

Before Report触发器的触发是在报表对象的任何格式化之前和在为报表查询数据之后。可以用这个触发器进行初始化设置。如果这个触发器失败，那么 Oracle Developer显示一个错误信息并返回。

After Report触发器在打印完报表或退出预览器以后启动。可以在运行报表以后利用这个触发器来清除那些不希望留下来的任何东西。即使在报表中触发器失败，这个触发器也总是启动。应该明显地显示某些错误信息，因为当这个触发器失败时，它不做任何事情。

可以利用Between Page触发器定制页面的格式。在Live Previewer中，Oracle Developer只在第一次显示报表页时才启动这个触发器。如果这个触发器失败，那么 Oracle Developer停止格式化页面并显示一个错误信息。

Before Form和After Form触发器用来操作报表的参数。不能从这些触发器中操作数据模型对象，比如列。不管是否取消运行时参数表单， After Form触发器总要启动。如果这些触发器失败，那么Oracle Developer停止执行这个报表。

China-pub.com
下载

注意 在Before Report、After Report、Between Page触发器中应避免使用DDL和DML语句。当Oracle Developer在开始运行建立报表时，它提取表的快照，如果执行 DDL和DML语句，这个快照将不再反映正确的数据词典，而且这可能会引起难以理解的运行时错误。

2. Format(格式化)触发器

当格式化一个实例对象时， Oracle Developer调用在报表中与该对象联系的Format触发器。可以改变对象的边界、和格式化掩码，可以格式化那些通过 SRWPL/SQL库程序包访问的对象的任何其他方面。可以跳过在基于变量值的对象内的处理。

如前面所提起的注意那样，在这些触发器中，应该避免 DML和DDL语句，否则这些触发器的启动可能是不可预测的。在单个对象的格式化期间，这些触发器可以启动许多次。

通过在域的周围放一个框架对象，而且把 Format触发器和框架联系起来，可以格式化矩阵中的单元。

3. Action(作用)触发器

在Previewer中运行报表时，为了能够做一些交互操作，可以在报表上放一个按钮。当这个按钮被按下时， Action触发器运行。要在分隔的 Previewer上运行另一个报表可以利用这个触发器，让你根据当前的报表值往下操作。有关的细节和其他特殊用途的报表请参看第 7章。

5.1.3 图形进程和触发器

Oracle Developer的图形进程也比表单进程简单得多，用图形作的大部分工作，特别是基本的显示对象的格式和数据库数据的联系，都是用 Layout Editor、查询和图形类型的组合来说明的。有关创建图形的细节请参看第 8章。有一些特定的操作，特别是涉及下钻 (drill-down)图形或条件判断的操作时，必须在图形触发器采用定制编程方式。图 5-16展示了图形进程，它包含几个触发器点。

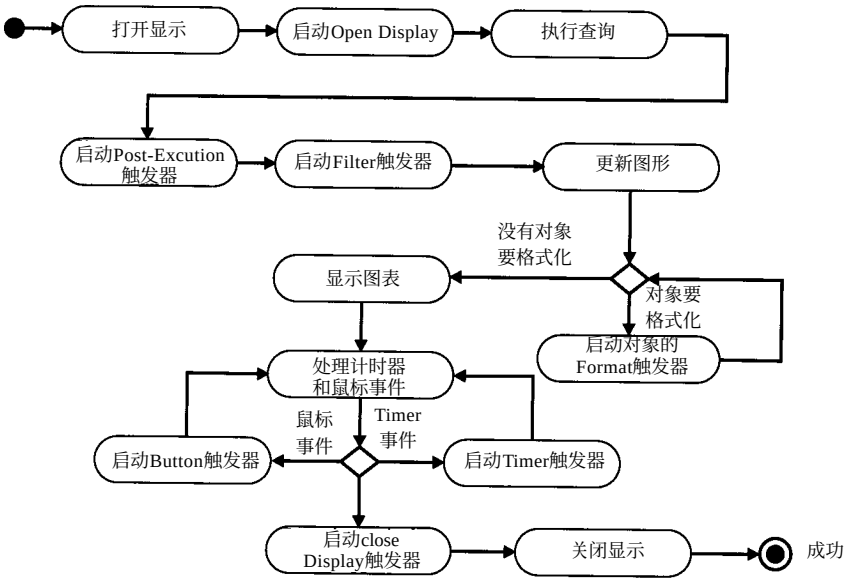


图5-16 图形进程活动图

China-pub.com 下载

在利用运行时图形系统打开图形显示时，为了显示而启动 Open Display触发器，然后进程执行查询，并启动Post-Execution 触发器，然后启动Filter 触发器，进程更新图形，启动那些和图形的对象有联系的 Format触发器。如果有计时器，那么在设置时间到时，计时器启动Timer触发器。如果在对象上有类似于按钮动作的鼠标事件 (例如针对下钻(drill-down)关系)，那么进程启动Button触发器。

可以建立用户自己的图形样板，作为编程格式的一种专用方法而在多个图形中使用它。可以利用各种触发器给特殊的图形显示增加专门的效果。

图形触发器

所有的图形触发器都是 PL/SQL程序(不像表单中的无记录的块)。可以通过拥有触发器的对象的属性表把任何程序连接到Object Navigator的触发器上。有专门的库程序包OG，它包含有一些函数，其功能是用于获得内部元素 (显示、查询、计时器等等)、格式化对象和执行不同的图形进程。

如果需要初始化图形上的某些东西，比如建立显示层、启动计时器或执行专门的查询，可以用Open Display触发器做这些事情。 Open Display触发器和Close Display触发器是Display Obiect属性表中的登记项。

如果想用某种方法处理查询数据，或者根据已有的查询建立某些东西，那么可以利用Post-Execution触发器。可以在Query属性表中建立它。

如果想根据其他的元素或查询的数据关系来格式化个别的图形元素，那么可以使用和这个图形元素联系的Format触发器。 Oracle Developer在元素的显示前执行Format触发器。可以从对象的属性表创建Format触发器。

Timer处理这样一些情况，比如想在某一时间间隔之后更新显示。一个普通的例子是，查询实时事务数据以得到全新的图形，显示出在一段时间内所发生的变化。 Timer有点复杂，应该用它们作一些试验来熟悉它们的性能。通过把 Timer触发器和查询或Timer对象联系，可以自动执行查询和/或执行Timer触发器。

按钮过程更复杂， Oracle Developer为向下钻数据的操作自动建立相应的事件和处理。但是如果想要更复杂的事件和图形变化，那么需要把 Button触发器和图形对象联系在一起。Oracle Developer处理鼠标移动事件(鼠标下、鼠标上、向下移动，向上移动)，可以设置对象接收这些事件。可以在设置后测试触发器程序中的事件。例如，可以在按钮图形中建立一个按钮进程，它断开计时器并自动更新显示。可以通过对象的属性表建立事件接收和创建按钮程序。

最后，还可以利用Close Display触发器清除所需要清除的任何东西。如果图形编程的其他部分发生了变化，你可以在这里把变化提交给数据库。

现在已经熟悉了Forms、Reports和Graphics的基本进程。本章的其余部分，讲述在使用它们时所要面对的一个基本的设计问题。虽然在所有的平台上，进程都是相同的，但 Oracle Developer的其他方面不一样。下一部分介绍跨平台设计的问题。

5.2 对因特网和其他平台的设计

基本工作做完以后，为了使应用程序可以在任何地方运行，还需要做些什么呢？有可能希望在不同的因特网浏览器、 Windows、X-Windows或Macintosh上使用应用程序，如何能保

China-pub.com
下载

证应用程序在所希望运行它的所有不同的浏览器和用户接口上都能运行得很好呢？跨平台开发有几个问题要考虑：

- 控件和菜单的标准：在目标平台上像按钮那样的控件的标准和菜单标准是什么？是否要完全遵守所有这些标准？跨平台操作的一致性比遵守平台的标准更重要吗？
- 颜色及模式：在不同的平台之间如何转换颜色和模式。
- 字体：对不同的平台如何转换字体？每个平台上的标准字型是什么？在应用程序使用的不同的监视器和打印机上的字型如何显示？
- 平台特有的功能度：可以利用任何图标、用户出口以及平台特有的特性比如 OLE或AppleEvents吗？要通过ORA_FFI使用内部的DLL吗？要使用操作系统的文件路径名特性吗？需要执行外部程序实现应用程序的某些特性吗？要使用本地平台上下文敏感的帮助吗？

注意 如果需要运行采用字符模式的应用程序，那么和以上的考虑有所不同。据Oracle公司称，在以后的版本中Oracle Developer不再支持字符模式。如果还没有安排把传统应用程序移到图形用户接口上来，那么现在是开始做的时候了。

你将发现通过计划和标准化来处理这些问题比当这些问题出现后再作处理要容易。

5.2.1 控件标准和菜单标准

每个平台都有专门的图形用户接口准则，所有的应用程序都应该遵守。 Oracle Developer自动地把标准的控制对象(按钮、复选框、单选按钮及其他)转换为适合于应用程序的设计平台的格式。

对每种平台的特殊准则请参看以下的参考资料：

- Windows 95/98 and NT4：Microsoft Corporation, The Windows Interface Guidelines for Software Design，Microsoft Press 1995.
- Macintosh：Apple Computer, Macintosh Haman Interface Guidelines，Addison-Wesley, 1992.
- Motif：Open Software Foundation, The Motif Style Guide Release 1.2 Prentice-Hall, 1992.

就我所知，没有因特网应用程序可接受的标准设计指南，因为因特网上装满了你可以想象到的任何一种设计。其他平台可以有它们自己的模式指南，你可以得到这些指南并把它作为参考资料。

现在，面临第一个大的决定是给定这些标准，你想遵守这些标准吗？通常答案是要遵守，但也不总是这样。如果你的用户要自己跨平台工作，那么应用程序的用户可能要从一个平台移到另一个平台，他们通常希望应用程序在每个平台上都一样工作。另一方面，如果用户总是使用同样的平台，但不同的用户却使用不同的平台，那么可能希望遵守平台标准，而让应用程序在不同的平台上显示出不同的特性。

下面用菜单、按钮和图标作例子。

在Windows中，MDI应用程序在MDI窗口提供单个菜单。在Motif中，每个窗口有它自己的菜单。在Macintosh中，在屏幕的上面只有一个面向所有应用程序的菜单。如果选择使用平台的标准，那么，为了与你的工作相适应，将需要一个单独的 Windows应用程序来支持MDI

China-pub.com 下载

方法，而对Motif和Macintosh则需要额外编码，使得能在每个窗口上正常地工作。另一方面，如果希望所有的系统都能相同地工作，那么可以在 Motif和Macintosh中进行修改，效仿MDI应用程序菜单的特性。

不同平台上的按钮差别很大。有两种基本的差别：护城河 (moats)和适航性(navigability)。在开始设计“吊桥”(drawbridge)之前，设计按钮要考虑的重点是一个“护城河”，这是一个缺省选择，它是在按回车键时执行。而适航性是利用键盘从控件移到控件的能力。 Windows的“护城河”与Motif和Macintosh的“护城河”相比是非常小的。当你从一个平台移到另一个平台时，按钮可能看起来实在奇怪。如果你的目标是坚持平台的标准，那么你必须对每个平台有不同的按钮设置。有关操作系统和图形用户接口访问信息的细节请参看后面部分的内容。

最后，图形说明了一个基本的事实：一幅图片可能胜过成千上万个字。但是不必花成千上万小时的努力。没有一个标准的图标格式。 Windows有它的ICO文件， Motif使用它自己的格式， Web页则使用GIF文件，而Macintosh把图标编译成应用程序。如果要使用图标，那么对每个平台都需要有单独的图标库，如果决定使所有的工具条和图标的按钮都相同，那么必须努力工作，使每样东西都一致。

5.2.2 颜色和模式

颜色，即使是一个点，也可能使应用程序变得活跃起来。但是使用太多或错误的颜色组合，也可能会把事情弄得一团糟。应用程序开发者近来经常会得到有关他们开发的应用程序的颜色出错的报告。当应用程序是在单色监视器上运行，或当使用者试图打印屏幕的图像时，或在屏幕上看到的关键组件出现的不是正宗的深兰色而是黑色的时候，尤其如此。如果一个平台使用标准不同的颜色，那么可能看不到类似于进度条那样的控件，因为它们可能使用了和操作系统控制的背景相同的颜色。另外，重要的是要知道许多 Motif和Macintosh用户用的是灰度级监视器，根本不容易区分颜色。所以，颜色不应该表示信息。

除非是丰富经验的颜色专家，否则限制在目标平台上使用将三、四种基本颜色混在一起的颜色和在单色或灰度设备上工作。把这些放到被命名的属性和样板上，确保所有的应用程序用正确的效果启动。测试所有目标系统上的颜色组合，确保它们工作。

同样的规则也适用于模式：不要滥用它们，因为在不同的显示中它们可能会有显著的不同。

最后，虽然Oracle Developer提供了一个256种颜色的调色板，但在个别显示中看到的实际颜色可能有所不同，有时差别还很大，特别是在 Windows中，这可能是一个实际问题。监视器的分辨率和驱动程序对颜色调色板也有影响。对于控件和窗口， Windows通过Control Panel的桌面颜色设置来控制颜色调色板。再次提醒你，限制你的颜色范围，并在目标显示中测试它们以发现问题。

5.2.3 字体

在平台之间转换字体仍然是相当困难的。 Oracle Developer做了它能做的一切，但最终它依赖于为可移植性所规定的转换方式。

所应该做的第一件事是建立 Coordinate System和栅格。理想的情况是把这些设置放在样板里，或者在创建新表单时建立它们。这个设置以后的变化可能会在所建立的布局中引起问

China-pub.com
下载

题，因为Oracle Developer用新的计量单位重新计算坐标。为了在平台之间有最大的可移植性，应使用厘米或英寸表示实数坐标。也可以使用点，它提供比较文本和控件的大小的能力，因为Oracle Developer总是用点来表示文本的大小。我发现使用英寸比较容易操作，因为我习惯于这种计量。然而在Windows中，它允许不同的屏幕分辨率，当在分辨率差别很大的监视器上查看应用程序时，一英寸的差别并不很大。为了设置这个选项，选择表单对象并显示它的属性表。找出Coordinate System属性，并在More按钮上单击，显示Coordinate Info对话框(见图5-17)。

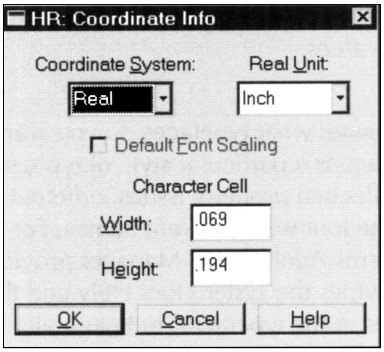


图5-17 Coordinate Info对话框

把Coordinate System属性设置为Real，把Real Unit设置为英寸或厘米，如果使用象素将使得在系统间移植应用程序很困难。现在，在每个画布中需要建立栅格。对于一个很细的栅格，把它设置为1/4英寸，在每个单元中有四个归整 (Snap)点。这就意味着可以把背景文本调整到一英寸的1/16以内，这对于可移植性来说，是一个适当的误差余量。要建立栅格的标尺，应在Layout Editor中显示画布，并选择Format |Layout Options | Ruler菜单项，选择这个项显示Ruler Settings对话框(见图5-18)。

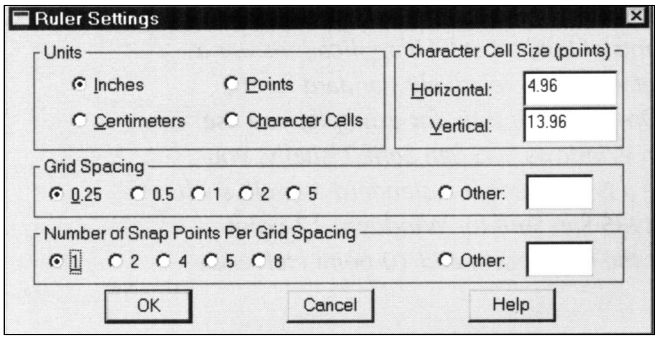


图5-18 Ruler Settings对话框

把标尺的Unit (单位)设置成与所选择的Coordinate System同样的单位(英寸或厘米)，把Grid Spacing (表格空间)设置为0.25，把Number of Snap Points Per Grid Spacing (每个表格空间的抽点数目) 设置为4(每1/4英寸4个抽点)。

提示 也可以使用Character Coordinate System。但由于字符的大粒状性质可能导致在画布中调整项和背景文本有困难。如有可能，通过项的Prompt (提示) 属性来使用提示说明，在其中规定调整的属性值。

现在选择字型调色板：想使用什么样的字样？哪种类型的大小和粗细？字样是一种特殊形式的类型。比如 Times Roman或Helvetica？因为智力属性问题，不同的系统和计算机销售商用不同的名字提供本来是相同的字型。例如 Apple Computer和Adobe Systems Adobe Type Manager提供Arial。而Windows 3.1提供系统字型Helv和True Type字型Arial。字型有较小的差别，在开发数据库应用程序中并不重要。但对于像 World Wide Web页面那样的应用程序来说，它需要表示特色和创造性字型，就变得比较重要了，是不是？好了，如果你想开发一些有创造性的应用程序，试一本像“停止偷羊”那样的书， Erik Spiekermann和E.M.Ginger著

China-pub.com 下载

(Mountain View, CA:Adobe出版社, 1993)但是，要认识到“创造性的”和“可移植的”可能是相互不相容的，至少就字型技术而言是这样。

注意 除了像字处理程序，或者是显示大量文本的其他系统那样的应用程序以外，大多数GUI应用程序使用对平台相对标准的简单字型。例如，在 Macintosh上使用Geneva，在Windows上使用MS San Serif。通常你还希望使用一种标准的类型尺寸，比如对Windows，用10点MS San Serif，对Macintosh用12点Gevena，在Motif用 10点Helvetica。

还应该决定在什么地方和如何使用黑体字、斜体字和下面加横线的字型。大部分时候应该尽可能少用这些字型特性，应该使不同显示文本对象的类型尺寸标准化。例如，使所有的标号为10点。如果需要在不同的平台转换字型的话，这将对你有帮助。做这项工作的最好方法是使用可视化属性Named和划分子类对象库的对象。有关使用 Oracle Developer的这些再用的特性的细节请参看第10章。

有可能发现，就像颜色在不同平台上不同一样，由于字型技术的不同，一个给定的字型在不同的平台不一致。应该在所有的目标平台测试你的标准字型。

可以在系统登录中的变量 FORM60_DEFAULT_FONT为应用程序表单的样本文本设置缺省的字型。在命名的可视化属性和属性种类中执行你的标准，确保应用程序遵守标准。

如果需要在平台之间转换字型，那么可以利用 Oracle Developer字型别名文件(在Windows中是UIFONT.ALI，在Macintosh系统是Preferences文件夹的Oracle Font Aliases 文件)。这些文件通过名字简单地把一种字型映射为另外一种字型。有关平台字型映射的细节，请参看目标平台的安装指南，实验不同的字形映射，直到对应用程序的效果满意为止。不同风格画布项要求不同的字型变换。

会发现在长久的运行中使用对象库来管理平台的差异比较容易，特别是如果要对不同的对象使用几种不同标准的字型时更是如此。有关使用对象库的细节，请参看第 10章。

5.2.4 平台特有的功能性

有几件可以在Oracle Developer上作的事情，限制在单个平台上作：

- 菜单和状态条/控制台。
- 图标。
- 用户出口。
- 平台特有的特性。

每个平台对菜单都有它自己离奇的处理方法。在 Windows平台上，MDI要求属于MDI窗口的单个菜单，该窗口包含所有其他的窗口。 MDI窗口还显示控制台或状态条。 Motif窗口(和Web窗口)是独立的，不包含在一个大的窗口内，每个窗口有它自己的菜单，可以从连接的父窗口得到。Maintosh窗口没有菜单，它们使用在屏幕上面的单个菜单，当把焦点转换到窗口时，它用窗口专用菜单代替上面的菜单。有关更详细的信息请参看第 6章“菜单”部分的内容。

图标来自图标文件，图标文件的格式与平台有关。所以对应每种平台需要建立适合那种平台格式的图标文件。 Windows有它的ICO文件，Motif使用它自己的格式，Web页使用GIF文件，而Macintosh把图标编译成应用程序。

在每个平台上用户出口被编译成平台可执行的库格式。如果应用程序依赖用户出口，那么将需要把代码从一个平台转到另一个平台。

China-pub.com
下载

最后，有一些可以在各个平台上利用的东西，但在其他平台上无效：

- Windows：DDE、OLE、VBX和Active X。
- Macintosh：AppleEvents。
- 所有的：通过HOST内部程序调用的任何东西，还有操作图形用户界面对象的用户出口。
- 因特网：以上所有的内容，均因为标准的 Java不允许使用这些特性，但其中 Active X将来可能会变化。其他的 Web限制：没有组合框 (Java没有和它们相当的东西)，没有GIF格式的图标，没有When-Mouse-Enter，When-Mouse-Leave和When-Mouse-Move触发器(要求的网络信息通讯太多)。还应该避免使用计时器触发器，避免大量使用图像，两者都可能引起因特网应用程序的网络性能问题。

提示 和Oracle Developer一起的联机文档资料广泛地讨论因特网应用程序的开发，在设计因特网应用程序之前一定要参考这些资料。

可以在触发器代码中运用的一个技巧是引用确定可移植性问题的应用程序属性。这让你通过使用Get_Application_Property内部函数检查这些属性的值来选择运行时使用哪些属性值：

- DISPLAY_HEIGHT和DISPLAY_WIDTH：(显示的高度和显示的宽度)这些这是告诉你当前的显示有多大，单位取决于是如何建立表单 Coordinate System的，这是表单模块的一个属性。
- OPERATING_SYSTEM(操作系统)：这个变量告诉你应用程序当前正在其上运行的平台的名字(MSWINDOWS、WIN32COMMON、MACINTOSH、SunOS、VMS、UNIX或HP-UX)。
- USER_INTERFACE(用户接口)：这个变量告诉你应用程序当前正在其上运行的用户接口技术的名字。(WEB、MOTIF、MACINTOSH、MSWINDOWS、MSWINDOWS32、PM、X、VARCHR2MODE、BLOCKMODE或UNKNOWN)。

5.2.5 标准和计划

为了可移植性必须做的一件事情是决定支持哪些平台。如果可以避免使用一些非常不同的平台，比如字符模式终端，那么会发现很容易达到可移植性。如果可以对一些应用程序进行一些限制，也可能有帮助。例如，如果销售汽车的领导因跟踪销售管理应用程序需要使用膝上型计算机，而其他人都使用 17英寸的SVGA终端，这样，只限制两个膝上机应用程序的窗口的尺寸就可以节省许多工作。在任何情况下，设计表单和图形的布局都需要知道窗口尺寸的变化。在某些情况下，也可以利用适当的滚动方式来适应较小的屏幕。请参看下一章的有关滚动画布和层叠画布的能力的内容。

一旦明确了所使用的平台，就把涉及到可移植性的用户接口对象和属性的标准放在一起。如果充分利用指定的可视化属性，属性类和图形样板能够使所设定的标准很容易在工作的应用程序中实现。那样一来，会发现把应用移到不同的平台是非常容易的。有关通过对象库、划分子类、指定可视化属性和属性分类(和其他方法)增加代码的可再用的细节，请参看第10章。

概括地说，关于可移植性问题，利用 Oracle Developer创立可移植的应用程序相对来说比较容易，但是要制定计划和标准才能有效和高效地工作。

现在已做好了准备，通过利用 Oracle Developer的高级编程和开发特性把实验模型带到下一阶段。接下来的部分研究更高级的问题，它们将显著增加创造安全稳固的应用程序的效率。

China-pub.com
下载

第6章　先进的窗口和菜单

一旦了解了Oracle Developer的Form Builder组件的处理过程，就得到了一种结构，可以在这种结构中扩展表单应用程序的范围。另外有两种要用 Oracle Developer研究的问题，可以使你设计大规模完整的应用程序的能力更加圆满。这就是窗口和菜单。用 Form Builder的这些特性来设计，只须很少或完全不用编程。

在第4章中，实验模型的表单只在单个窗口上呈现单个画布。本章讲述如何利用多个画布把应用程序扩展到多个窗口，这些特性为提供建立对话框、工具条和警报的能力。

在第4章中，应用程序的缺省菜单不包括在用户可以利用的范围中。用户可以建立自己定制的菜单，把表单的报表、图形或想从表单中调用的无论什么其他的东西结合起来。

6.1　窗口

在Form Builder中制作窗口意味着要作两个事情之一：创建窗口的内容或创建和管理窗口本身。下面的讨论扩展第 4章的内容，讲述如何充分利用 Oracle Developer画布和窗口在应用程序的设计中起重要作用。

6.1.1　描绘画布

回想在第2章和第4章中用画布来显示块和项，在Layout Editor中编辑画布，画布成了建立表单中所做的许多工具的基础。那么确切地说，画布是什么？

画布是摆放样本文本和项目的背景。每个项目在它们的 Property Palette上都准确地涉及一个画布。可以在不同的画布之间分配块的各个项目。

画布并不是孤立地作为一个界面对象。要观看画布和它的项，必须在窗口上显示画布。Oracle Developer使这些独立的对象为你建立窗口，从而为画布提供一个视图：包括部分画布

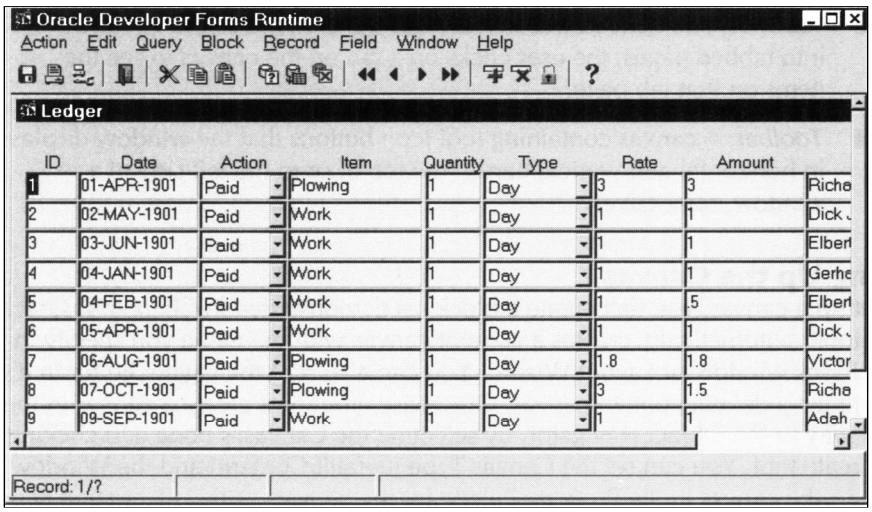


图6-1　Ledger应用程序的一个窗口

China-pub.com
下载

或全部画布的长方形。例如图 6-1显示Ledger应用程序的一个窗口，这个窗口显示了基础画布的一大半。可以通过窗口看到的这部分画布叫视图。窗口有水平滚动条和垂直滚动条，可以通过滚动画布来观看视图。

这样把显示方式和显示内容之间的工作划分成独立但是相当的两部分工作，使建立表单与显示它们的窗口的大小无关，这样作加大了视觉范围但却缩小了界面。你会发现用户对于需要不断滚动通过画布的界面不满意。好的界面应在所有的时间显示画布上的所有域和项。但是，就像生活中的大多数状况一样，与能够同时显示项比较，有更多的需要。窗口和画布提供了多方面的适应性。

1. 选择适当的背景

为了得到多方面的适应性，首先要考虑画布所使用的类型。某些画布不只是可以在窗口内滚动，还可以做更多的事情。有四种类型的画布：

- 内容画布(Content)：包含窗口“内容”的画布，每个窗口至少有一个内容画布，而且通常只有一个。
- 层叠画布(Stacked)：在另一个画布上面显示的画布，通常包括和基础的内容画布上的项分开的组内的一些项。
- 标签画布(Tabbed)：一种内容画布，它把窗口的内容分成指定的页，用户单击画布的标签页观看在那个标签页上的项目内容。
- 工具条(Toolbar)：包含工具图标的画布。分别在窗口的顶部或左边上的水平和垂直条上显示这些图标按钮。

2. 填充画布

为了建立一个画布，首先要创建一个块，就像在第4章中说明的那样。Oracle Developer用在New Block窗口或Layout Wizard中指定的名字自动创建内容画布。也可以在新表单中启动Layout Editor得到缺省的内容画布。否则，必须在Object Navigator中用通常的方法，通过选择Canvases结点并单击Creat工具来创建画布。可以在画布的属性设置板上设置 Canvas Type属性(缺省设置是Content)，显示画布的窗口。像图6-2显示的那样。

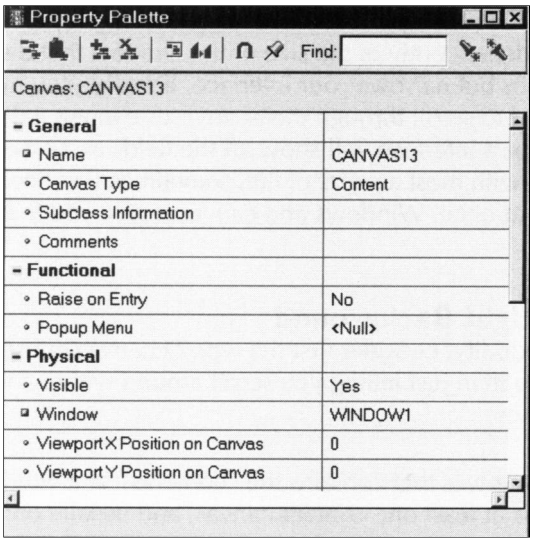


图6-2 画布窗口

要查看Layout Editor中的画布，双击这个对象的Object Navigator图标。第4章讲述过如何利用Layout Editor把项目和样本文本放到画布上，这里唯一的区别是由 scratch(擦除)建立画布，而不是用创建块时 Oracle Developer产生的一组项目的画布去工作。可以在 Object Navigator中创建一个项，然后把这个项的 Canvas属性设置为新画布。也可以在 Layout Editor中创建项目，并通过它的Property Palette改变这个项目的属性。可以创建第二个画布，并通过改变已经存在的项目的 Canvas属性把项目从基础画布移动到它的上面。如果你把这个属性设置为NULL，那么这个项将不出现在画布上，这可以定义一个项目不显示它的值。

China-pub.com 下载

可以通过View | Show Canvas菜单控制画布背景的显示。当Layout Editor有效时，可以得到这个菜单项。如果将这个菜单项设置为 off，那么将直接在项目下的背景上看到网格。如果把它设置为on，那么将看到没有网格的画布。其次，如果想重新决定画布的大小，可以通过拉动画布右下角的选择手柄来做。要了解这个过程，可接通 Viaw |Show Canvas菜单项，然后在画布的右面或底部边缘上单击。也可以改变画布属性设置板 Width属性和Height属性。

使用View | Show View菜单可以观看视图。如果在画布的窗口上显示画布，这个视图就是所看到的画布的区域。 Layout Editor把视图显示为带有手柄的一个黑色的长方形，可以利用这些手柄来重定缺省窗口的大小，也可以通过改变窗口 Property Palette上的窗口的Width和Height来改变这个尺寸。通常，这项工作是把窗口的宽度和高度设置得和内容画布的 Width和Height相同，这样可以使窗口正好包围画布。也可以通过拉动这个长方形，或通过改变画布的X | Y Position属性来移动视图，这样，可以把画布定位在窗口内的某个位置上而不是在它的左上角。

注意 如果是层叠画布，则在画布的边界有另外的视图。可以在层叠画布上放一个滚动条并在窗口内滚动它。“层叠画布”小节会给出更多的细节。

可以利用所有这些特性来放置内部项和它们的画布，在显示它们的窗口中取得最好的效果。例如，通常希望窗口紧紧地包围着画布，而且希望给出的画布的项目布局尽可能小。如果可能，希望用户不用滚动就可以看到窗口内的所有的项，使表单使用容易些。为此，重新把画布的尺寸定为正好包围项的布局，然后把画布的 Width属性和Height属性拷贝为对应的窗口的Width和Height属性。这样，就把窗口的尺寸精确地定为画布的尺寸了。

用户可以通过选择一个边缘并拉动它来重定画布的尺寸，这就是说，可以为了屏幕显示而调整窗口和画布的大小。可以确定一个初始窗口尺寸，从画布的一边滚动到另一边，以适合较小的视频显示。而用户可以用高分辨显示重定画布的尺寸，使得可以同时看见每一样东西。

通过选择画布并单击Delete 工具，可以删除Object Navigator上的画布。通过把它们的画布属性设置为NULL，Oracle Developer可以清除任何与被删除画布有关的项。

在显示窗口时，自动显示窗口的基础画布。可以通过触发器中的 Show_View和Hide_View内部子程序控制其他画布的显示。

3. 层叠画布

层叠画布允许产生几个特殊的效果：

- 可以在背景的独立簇中创建一组按钮或其他项，用图形把簇和基础内容画布区分开。
- 可以创建可分离和可重用的项目组，比如按钮组。通过复制，可以在不同的画布上再使用(第10章讲述细节)。
- 可以有计划地隐藏或显示层叠画布，创建一个当用户执行某个动作时自动变化的视图。
- 可以在多个动态变入变出的内容画布的上面显示不变化的文本和域。有一种方法，不用创建很多个项就可以有重复的元素。
- 在滚动基础的内容画布查看其他项目时，可以让在多记录显示中的一组项目留在屏幕上。例如，当通过滚动检查其余列时，可以在改变的列中显示主关键字，使得不管在画布的什么地方总可以看到主关键字，这个特殊效果就像电子表格中的多显示视图一样。

China-pub.com
下载

在主-从表单中，当在合适的窗口上同时要看的项太多时，可以使用一个技巧，但是可能帮助不会太大。例如，如果目标显示平台是一个 640×480 VGA的Windows显示器，而表单和Ledger表单类似，那么，将没有办法让用户可以同时看见具有全部的域的记录。 Windows采用滚动可以解决这个问题。把应用程序改为显示面向人员的 Ledger内容，显示给定的人员的所有Ledger项，个人的信息在应用程序的顶部，而在它下面显示 Ledger信息多个项的多个记录。如果有滚动，则当滚动屏幕时，就会丢失个人的信息。而滚动层叠画布可以帮助应付这种情况。

1) 单击Create 工具，创建Person数据块，利用Data Block Wizard并且只显示Name列。在Layout Wizard中创建一个新的Content画布，并使数据块的类型为Form。

2) 在Object Navigator中选择Person数据块后，在Create工具上单击，创建Ledger数据块。利用Data Block Wizard添加所有的Ledger列和Person.Name查找列。

3)在Ledger数据块的Layout Wizard中创建一个画布作为新 Canvas，并把Type 设置为Stacked (层叠的)，把数据块类型设置为Tabular ，并用滚动条使它包含5行数据。

4) 把新表单作为Person_Ledger保存。

5) 导航到Object Navigator中的Person块Relation结点上和(Person_Ledger | Data Blocks | Person | Relations)并单击Create工具创建主-从关系。像在第4章那样根据Ledger和Person之间的对象关系创建关系。

6) 在Object Navigator结点上双击，打开Layout Editor中的Person画布视图。

7) 为了帮助看清画布之间的关系，选择View | Stacked Views菜单项。显示Stacked/Tab Canvases对话框(见图6-3)。提示选择要显示的层叠画布。这个对话框列出所有的层叠画布并让你用一种不平常的方法接通和判断它们。代替用顺序的单击或双击操作不断地点击画布，用单击画布来进行选择画布，用CTRL-单击操作来淘汰画布。就是说，在层叠画布名字上单击来显示它，并在它上面CTRL-单击操作从显示中除去它。

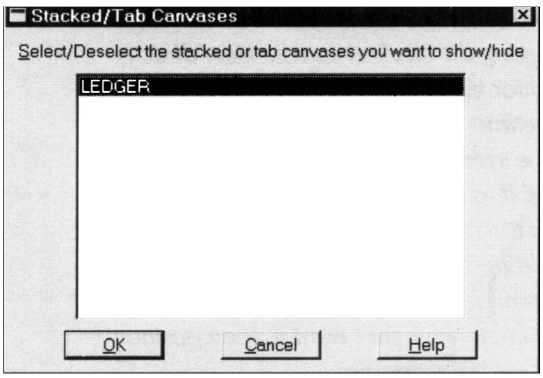


图6-3 Stacked/Tab Canvases对话框

8) 单击OK，在Layout Editor中显示或删除层叠画布。对于Person-Ledger表单，所得到的Layout Editor显示出的Ledger块完全遮住了Person块。

层叠画布视图是画布的可见部分，虽然内容画布的视图与窗口的边界有关，但层叠画布的视图与窗口没有关系。可以通过在 Object Navigator中选择画布，查看层叠画布的边界，然后显示Layout Editor。在层叠画布的边界上可以看到控制尺寸的手柄，而且在内容画布的标尺变成灰色时，画布的标尺是活动的。拖动手柄改变视图的大小以显示更多或更少的层叠画布。

定位层叠画布最容易的方法是在画布的 Property Palette中设置视孔坐标。

9) 在Object Navigator中选择Ledger画布。

10) 通过选择Tools | Property Palette菜单项显示弹出菜单上的Property Palette，或通过右

China-pub.com 下载

击画布对象，显示属性调色板。

11) 找出Viewport header和Viewport Position两个属性。把Y位置设置为0.8英寸，把层叠视图放在Name 项的下面。图6-4显示了在Layout Editor中的结果，可以看到利用明暗区别的层叠画布标尺。

注意 不要混淆出现在Physical属性组上Viewport Y Position和Canvas属性上的Viewport Y Position。后者定位观测孔中的层叠画布，不是观测孔本身。这是让你第一次显示层叠画布时，把滚动画布放置在不是最左边或顶部的其他地方。也可以在Layout Editor中直接拉动层叠画布视图，而不通过它的属性来设置它。有时候层叠画布的边界看不见，而且要找到它来拖动可能有点困难，可以在画布背景单击，然后寻找重定尺寸手柄，再单击定尺寸手柄并保持拉到某处。我通常喜欢拖动画布，直到它接近我希望的地方为止，然后通过设置属性再准确地定位它。

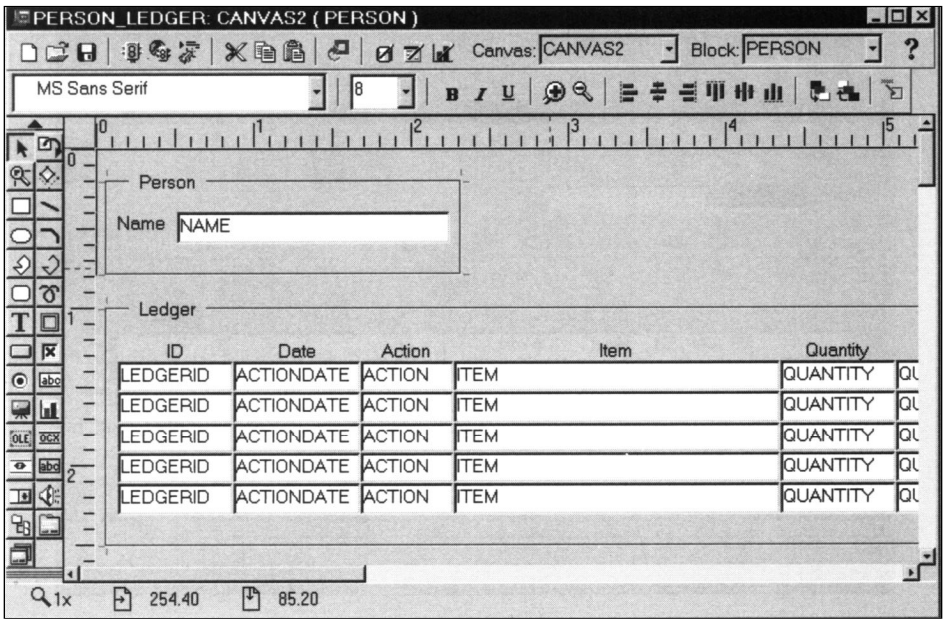


图6-4 在内容画布内显示的层叠画布的Layout Editor

12) 把Show Horizontal Scroll Bar(显示水平滚动条)属性设置为YES，使水平滚动条有效。

这个滚动条出现在视图底部边缘的下面，所以必须适当地确定视图的大小，使滚动条位于所希望的地方。在我们的例子中，这个视图将扩大到刚好在第五个 Ledger记录的最后一行的底部，滚动条正好出现在记录块的下面。在 Layout Editor中看到的滚动条是内容画布的而不是层叠画布的。

13) 最后，必须通过Layout Editor边界或通过画布和窗口的Property Palettes，确定内容画布的尺寸，使它包含它自己的项和层叠画布，必须确定窗口的尺寸使它包含内容画布。

运行这个例子所得的显示展示在图6-5中。

注意，在图6-5中有两个水平滚动条，下面一个是和窗口有关的滚动条，只有画布扩展超出了窗口时，这个滚动条才出现。在这里是刚超出一点点。上面的滚动条和层叠 Ledger画布有关，可以利用它在Amount域记录滚动条上滚动。

China-pub.com
下载

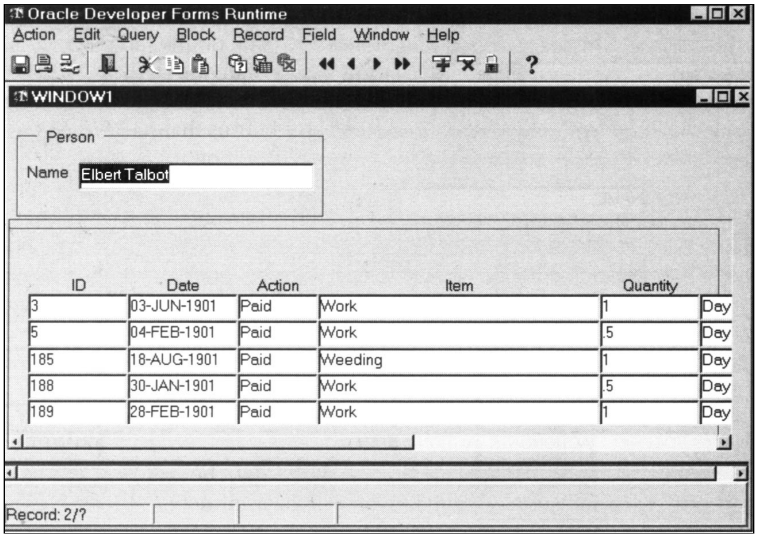


图6-5 Person Ledger表单和它的滚动视图

注意 如果你看不到层叠 Ledger画布，应检查两样东西。首先，确保 Person画布在 Navigator Lagout组中的次序是第一，第一个先显示，第二个画布在它的顶部显示。如果 Person画布排在第二，那么就遮住了层叠画布Ledger。其次，确保层叠画布观测孔把层叠画布框架放置在人员块框架的下面(查看兰色阴影线和框架手柄)。试着把观测孔往下移动一点点看会发生什么。如果层叠画布的项目遮住了内容画布的项目，即使是遮住一小点，那么在导航它时，内容画布自己升高，显示全体被遮住的项目。这将是显示画布之后表单做的第一件事。你会看见层叠画布出现，然后消失，就像是变戏法一样。

4. 标签画布

标签画布在一个或多个标签页中显示它的项目，标签页是一种微型画布，与标签文件夹类似。在标签的标记上单击， Oracle Developer就显示单击的那一页。利用标签画布可以按组为单位来排列项。这就允许显示某一组而隐藏另外一组，为用户简化了显示。可以利用这个特性建立有许多项目的表单，但可以用这样一种高级结构和控制的方法把它们提供给用户，使表单非常容易使用。不仅如此，而且标签画布对 Oracle Developer支持的所有图形接口都是完全可移植的。

第4章说明了创建标签画布的基本过程，这个部分讲述如何把标签画布当作层叠画布来处理。这样可以把一个主要内容的画布和一系列从属内容的画布结合起来。例如，第 4章中的 Worker Skill表单用在一个标签页上的人员和在另一个标签页上的技能，把一个人和一组技能连接在一起。这是一个缺点，在寻找技能时，要把人的内容从显示中除去。反之亦然。理想的情况是，用户同时可以看到人员的信息和技能的信息。

WorkerSkill表单的改型是更通用的HumanResources表单，它用主-从关系把住所与技能和人员连接起来。这个表单有三个数据块，对应于 Person、Lodging和WorkerHasSkills三个表，Person画布在内容画布中有Person的名字。在布置另外两个数据块时，不选择内容画布或层叠画布，而是选择带有一个Lodging页面和另一个WorkerHasSkills页面的独立的标签画布。第一次通过Data Block Wizard时，选择一个新的标签画布。第二次通过时，选择第一次通过时创建的画布，然后选择(New Tab Page)。Lodging画布有表单布局，而Skills页有表格布局，列出

China-pub.com 下载

不同的技能。

一旦完成了画布的布置，给定了尺寸和布置好它们的项目，就可以利用层叠画布组合Person画布和Ledger画布的方法，组合这两个画布。

1) 显示Person画布的Layout Editor并选择View |Stacked |Views菜单项。

2) 从Stacked/Tab Canvases 对话框的列表中选择标签画布。像前面一样，选择调整在内容画布项顶部下方的画布。

3) 选择标签画布并拉它向下，或改变它的 Viewport Y Position 为大约0.25英寸的值或稍大。

这就产生了显示在图6-6的布局和显示在图6-7的运行的表单。

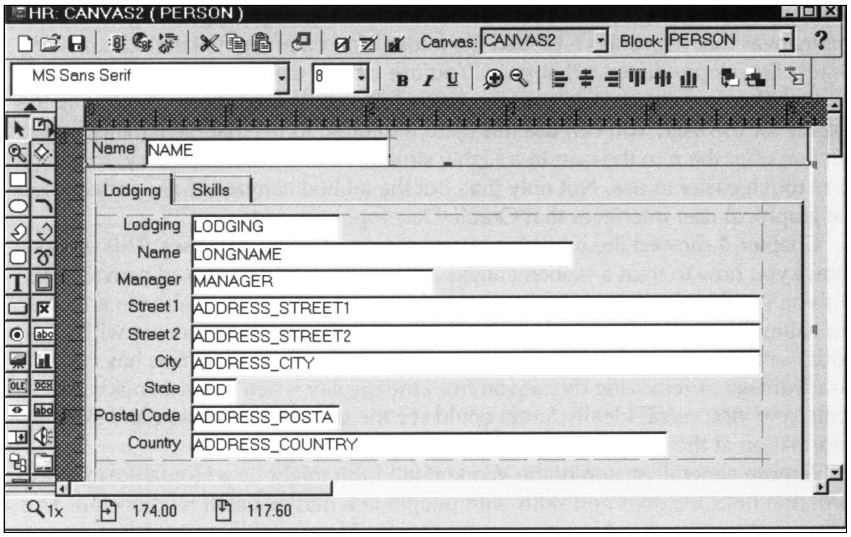


图6-6 带有层叠标签画布的Layout Editor

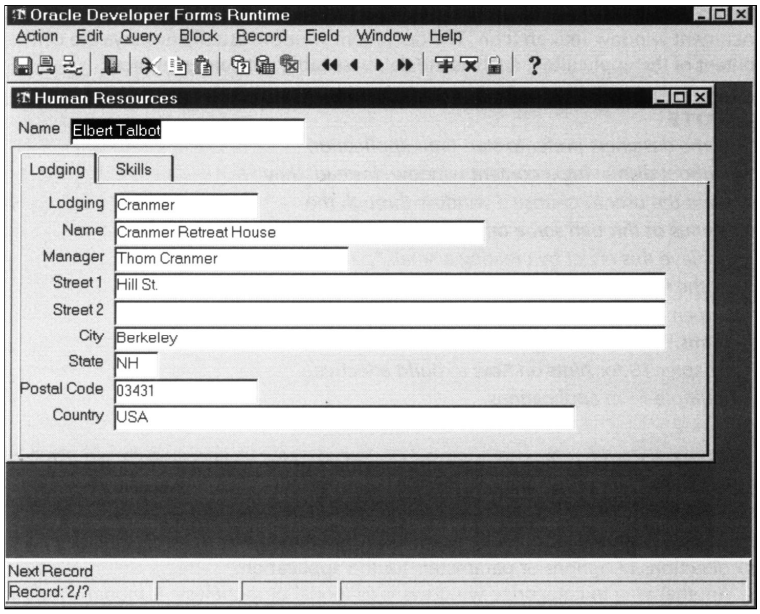


图6-7 运行时的标签画布

China-pub.com
下载

布置完画布以后，下一个步骤是通过窗口显示它。以下部分讲述创造性地利用画布的不同种类的视图，这些视图是可以创建的。

6.1.2 通过窗口观看

用画布组织表单的内容，通过窗口让用户看到它。通过在表单中添加画布和窗口，可以创建易于理解和使用的表单。每个窗口至少有一个定义窗口内容的内容画布。

1. 利用多种文档

微软Windows平台引入多种文档接口(MDI)，在应用程序中组织多个窗口的使用。Oracle Developer通过采用特定的假设和可利用的条件采用MDI体系结构并把它扩展到其他平台。

MDI应用程序有一个“应用程序窗口”(application window)，它拥有应用程序中所有其他的窗口。“应用程序窗口”没有画布，而是显示其他的窗口。主菜单属于“应用程序窗口”，而这个窗口总是打开的。这个窗口的子女有表单窗口，而且还可以是文件或对话窗口。

注意 除了Windows和NT以外，其他平台不使用MDI体系结构。Motif和Macintosh不采用它。

文档窗口(document window)是这样一种窗口，它完全被包含在应用程序窗口内。如果移动文档窗口超出了应用程序窗口，那么移动超出这个窗口的那部分文件将消失。通常，在应用程序窗口显示区域的一个重叠组或堆砌组上显示文档窗口。也可以把文档窗口增加到最大值，占去应用程序窗口的整个显示区域，或者把文档窗口减小成图标。文档窗口通常显示应用程序的主要内容，比如数据库表数据和图形。

注意 一些设计者喜欢启动他们的应用程序时不显示内容窗口。他们希望用户通过菜单或通过某些其他的界面来选择窗口。可以通过创建一个小的“父”表单作为运行表单来取得这种效果。这个表单可以有一个溅开的(Splash)屏幕图形或一组打开其他表单的按钮。有关如何建立有效的多表单程序请参看第10章“使用多重表单”的内容。

对话框是一个窗口，它与应用程序窗口无关。应用程序窗口不限制对话窗口，也可以把对话窗口移到应用程序窗口外面(如果它的属性允许这样的话)。对话框通常包含有让应用程序和用户交互的域，比如应用程序选项或参数的集合等。

窗口分类的另一种方法是分为模式的或无模式的。模式窗口(modal window)要求用户在应用程序的任何其他窗口做任何事情之前响应并消除这个窗口。因为对话框的主要功能是要得到往前进行操作所需要的信息，所以对话框通常是模式的。也可以有无模式对话框。如果这个对话框的信息不是以后处理的部分信息，那么在应用程序中做其他事情时，这个对话框可以继续存在。例如搜索字符串的Find对话框，就可以是无模式的。无模式窗口(Modeless window)是在移动到另外一个窗口时并不消除第一个窗口。通常模式窗口有显著的特性，比如没有滚动条、具有固定的尺寸、不能缩小为图标等。在某些平台上，模式还扩展到其他的应用程序。就是说，在系统中做任何其他事情之前，必须响应这个窗口并消除它，而不只是应用程序(系统模式与应用程序模式相反)。

在Oracle Developer应用程序中，应用程序窗口和当前执行的表单相对应。应用程序的主菜单是表单的菜单。通过指定它们的画布为表单的Horizontal MDI Toolbar属性和Vertical MDI Toolbar属性，也可以具有水平的和垂直的工具条。

China-pub.com 下载

通过使用内部过程Set_Window_Property中的名字FORMS-MDI-WINDOW，可以在触发器的编码中用程序设置应用程序窗口的属性。

2. 创建窗口

在创建新表单时，就得到一个新窗口。第一个画布将自动地使用这个窗口。如果想创建更多的窗口，通常的方法是使用Object Navigator选择Windows标题并在Create工具上单击。

利用在Window Property Palette上的属性，几乎可以确定用窗口做的每一件事。“功能”部分可以建立窗口的基本交互能力。展示在表 6-1的属性叫做“提示”，因为它们不是所有的都适应于所有的平台。

要建立窗口的基本结构，可以利用表6-2中的属性。

表6-1 功能Forms窗口属性

属 性	说 明
Close Allowed	是否可以用平台特有的关闭命令关闭窗口
Resize Allowed	窗口的高度和宽度是否固定，或用户是否可以重定窗口的尺寸，Resize Allowed设置为NO只在Maximize Allowed设置为False时有效
Minimize Allowed	用户是否可以把窗口缩小成图标
Inherit Menu	窗口是否显示表单菜单，在微软Windows中无效
Move Allowed	用户是否可以移动窗口
Maximize Allowed	用户是否可以用平台特有的操作最大化或放大窗口，只在 Resize Allowed为False有效

表6-2 结构Forms窗口属性

属 性	说 明
Primary Canvas	在作为主内容视图的窗口中显示的画布的名字。主内容视图是一个基础视图，当你打开窗口时，窗口总要显示它，这是可任意选择的，但是如果你使用Show-Window来显示窗口或者如果使用层叠画布并直接导航到它们那里，就可能是必须的了
Vertical Toolbar Canvas, Horizontal Toolbar Canvas	在垂直或水平工作条显示的画布，请参看“创建工具条”的内容
Window Style	窗口、文件或对话的种类
Modal	窗口是否是模式的
Hide on Exit	当导航离开一个无模式窗口进入下一个窗口时，是否关闭和使这个窗口不活动，只在Modal是False时有效
Direction	文本流的方向，Default (缺省)设置通常是所希望的方向，因为它从Oracle的NLS建立中继承正确的方向
Icon Filename、Minimized Title	图标文件的名字和当用户最小化窗口时显示的标题，只有当Iconifiable为True时有效。这个图标文件的路径和特性取决于平台，而且使用图标文件要求作额外的工作把应用程序停泊到其他平台
Show Horizontal Scroll Bar、Show Vertical Scroll Bar	是否要显示水平和垂直滚动条，如果画布比窗口大，可以在画布上滚动视图，只对无模式窗口有效

注意 当你导航到在当前窗口外面的画布上的项时，Oracle Developer自动地滚动窗口的画布，也可以使用Set_Canvas_Property内部子程序通过触发器，或者通过画布的Canvas属性上设置 Set_View_Property 的X或Y的位置来控制滚动。应当使用Set_Canvas_Property，而Scroll_View只是提供来用于Oracle Developer前面的版本向后的兼容性的。

China-pub.com
下载

Oracle Developer可以很容易地显示和关闭窗口，为通过导航它自动控制这个过程。当导航到没激活的画布上的项目时，Oracle Developer在它有关的窗口显示这个画布。当导航离开这个画布进入不同窗口的另一个画布时，如果把 Remove on Exit属性设置为True时，Oracle Developer自动关闭第一个画布，否则，这个窗口只是放到后面一层，但仍然显示。

注意 有关模态窗口的属性请参看以下部分的内容；Remove on Exit(在退出时除去)属性不适用于模态对话。

也可以通过用内部的导航子程序或用内部的 Show_Window和Hide_Window程序导航，在触发器中打开和关闭窗口。如果在当前活动窗口后面有一个静止的显示的窗口要这样做，那么Oracle Developer把这个静止的窗口提高到窗口堆堆栈顶部并激活它。可以用导航或用Show_View和Hide_View，采用同样的方法显示或隐藏特殊的画布。

3. 创建和关闭模式对话框

创建一个模式的对话框的步骤：

1) 把Window Style 设置为Dialog (对话) 并把Modal 设置为True。

2) 与它们的块一起创建内容画布和组成对话框的项，如果需要，把画布和对话框联系起来。

3) 固定尺寸对大多数对话框有利，所以把 Fixed Size属性或Resize Allowed属性(允许重定尺寸)设置为True，并把窗口的Width 和Height属性设置为画布的尺寸，确保窗口显示整个画布。

4) 使对话框比这个窗口或比它将要在其上出现的窗口小，这可以使用户知道应用程序的模式状态。

应该把模式对话框的显示建立在适当的事件上。做这项工作有两种简单的方法：第一，可以按照用户在对话的画布导航上的项的次序排列项目导航的次序。也可以不考虑导航的次序，用按钮触发器或菜单命令明确地导航。在任何一种情况下，当用户导航到对话的项目时，Oracle Developer自动显示对话框。

第二，可以从按钮触发器或菜单命令中调用 Show_Window内部子程序，这个程序显示窗口并导航到对话框画布的第一项。

Oracle Developer中的Modal属性防止用户用鼠标离开对话框导航。它不允许，但暂停(Suspend)按键允许。所以用户可以利用导航键导航出去。当用户这样做时，Oracle Developer自动地关闭这个对话框。然而这种特性不是用户在对话框所期待的。Modal对话框作为活动的窗口应该保持在屏幕上，直到用户用确定的动作消除它们为止，通常是通过在 OK或Cancel按钮上单击。

有几种方法关闭(断开)按钮导航键：

- 用对话框内的项把这些块的Previous Navigation Block属性和Next Navigation Block属性设置为这个块本身，使导航循环。如果这样做，应该有单独的块和所有出现在对话框画布上的项。
- 在拥有正在显示的项的块上创建Key-Others触发器，包含一个null语句：

```
NULL;
```

除了已定义了键触发器的键以外，断开了其他所有的键。需要最小限度地定义 Key-Next-Item和Key-Previous-Item以便允许在对话项之间跳格。

China-pub.com 下载

然后需要在OK和Cancel按钮上建立触发器，或用任何一种方法指示 Oracle Developer关闭对话框。例如， When-Button-Pressed触发器导航离开窗口时， Oracle Developer自动关闭这个对话框。必须要用在对话画布外面的块导航到无模式窗口的项。

可以从一个模式对话框中打开另一个模式对话框。例如，可以有一个 Options按钮，显示选项所扩展的第二个对话框。还可以达到某些奇特的效果，比如用一个 Extended按钮，产生第二个更大的对话框代替当前的对话框。自己试验作一下。

4. 创建工具条

工具条是沿着窗口顶部或左边的一条图标。要创建它，必须首先创建包含按钮项和样本图形的工具条画布 (画布类型是Horizontal Toolbar 或Vertical Toolbar)。可参看前面部分有关创建画布的内容。

1) 为画布指定将要显示工具条的窗口的 Window (窗口)属性。

2) 给工具条设置宽度和高度，设置值为所设计的应用程序最优的尺寸。例如，希望水平工具条的工具条画布的宽度和窗口的宽度相同，而高度正好包围图标。

3) 下一步是为窗口对象的Horizontal Toolbar 或Vertical Toolbar 属性设置为适当的画布的名字，告诉窗口在适当的位置显示这些工具条。

注意 在微软的Windows中，也可以通过表单对象的Horizontal MDI Toolbar或Vertical MDI Toolbar属性显示MDI应用程序窗口工具条。Oracle Developer对这些工具条忽略工具条画布的Window(窗口)属性。

单个窗口可以有多个工具条画布。 Oracle Developer使用标准的规则来显示它们。要显示它，只要导航到工具条上的第一项上，而不是原来的工具条上。通常，动态变化的工具条并不好，因为这种变化会使用户感到混乱。可以用不同的图标代替禁止工具 (例如，暗淡、灰色、十字画叉，等等)。可以利用这个特性禁止和允许工具条上的工具有效。

因为工具条上的项就是块上的项，必须进行一些专门的测量，确保这些项所起的作用和表单的其他项都不同，特别是有关导航的作用。通常，把工具条的按钮项作为单独的项比较好。把前一块的Next Navigation Data Block属性设置为是对应块而不是工具条块，把这个块从块导航中取出，可以用Property Palette去做这件事。

4) 对于每个项，把Property Palette中的Keyboard Navigable 属性和Mouse Navigate属性设置为NO。把这些属性断开是防止在工具条上的工具上单击引起游标导航进入工具块。当在工具上单击时，用户能够保证停留在他们所在的块内。

注意 如果不用用户自己的菜单代替标准的Oracle Developer菜单来运行MDI应用程序，那么将会看见在MDI窗口顶部的主菜单下面出现工具条。这是一个特殊的菜单工具条。缺省的菜单工具条包含有和普通菜单项对应的图标，比如Enter Query、Next Record等。不提供自己的完整的主菜单模块，便不能改变这个工具条。也不能移动它。有关增加定制菜单的细节，请参看本章后面的“菜单”部分的内容。

5. 创建警报

警报是一个模式窗口。它显示信息或询问简单的问题，要用户给出 Yes / No类型的响应信息。Oracle Developer有一个专门用于报警的对象，可以简化对那些非常普通的窗口的编程。

注意 当在PL/SQL代码中使用Message函数或当存在一个内部消息而且有多个信息时，

China-pub.com
下载

Oracle Developer自动显示警报。通常，消息出现在状态条中。如果有多条信息，则用户只能看见最后一条信息，并丢失临时的信息。而Oracle Developer把临时信息作为专门的警报显示。通过使用内部的Synchronize子程序，在调用内部的Message子程序之后，立即在PL/SOL代码中显示这些信息。有时候可以避免这些情况，比较好的方法是，对错误使用明确的报警对象去完全代替信息系统。例如，可以为标准的错误建立一组可再用的报警对象并把它装入标准对象库中。

要创建警报，在对象导航器上选择 Alerts标题，并用通常的方法在Create 工具上单击。有三种警报。通过警报属性设置板的 Alert Style属性规定使用哪种类型：

■ Stop (停止)：显示一个停止信息图标和信息 (见图6-8)。

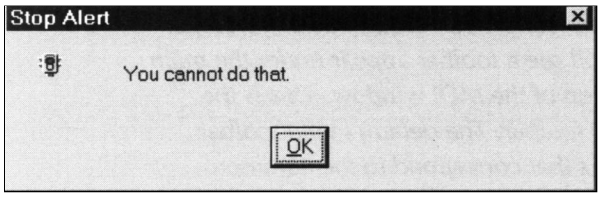


图6-8 Stop Alert对话框

■ Caution (警告)：显示一个感叹号图标和信息 (见图6-9)。

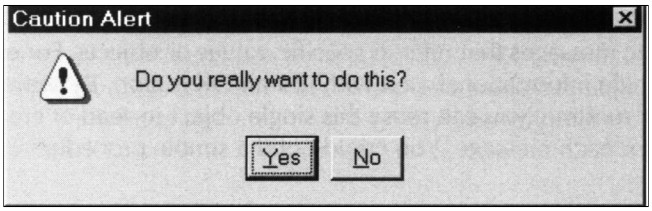


图6-9 Caution Alert对话框

■ Note (注意)：显示一个信息图标和信息 (见图6-10)。

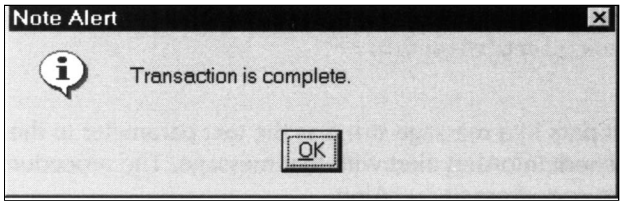


图6-10 Note Alert对话框

在Message属性中输入要在警报中显示的信息。可以输入多达 200个字符(虽然不是所有的平台都可以显示全部数目的字符)。

用Property Palette中的Button1、Button2和Button3 标明显示警报的按钮。如果不输入标记，那么警报不显示按钮。例如，如果使用缺省的 OK和Cancel 设置值，那么就得到带有“ OK ”和“ Cancel ”两个标记按钮的警报。如果删除 Button2的 Cancel设置，那么就得到只有“ OK ”标记的单个按钮。至少必须标明一个按钮。还应该指定一个按钮作为缺省按钮。当用户按ENTER键时，Oracle Developer选择缺省按钮。

利用Show_Alert(显示报警)内部子程序由触发器代码显示警报。这个子程序根据用户选择

China-pub.com 下载

了的按钮返回编号ALERT_BUTTON1，ALERT_BUTTON2或ALERET_BUTTON3，然后可以测试返回的编号，并在触发器代码中采取适当的动作。例如，下面的代码用 Caution警报提示用户决定是用触发器继续，还是不继续。按钮 1是Yes，而按钮2是No。

```
DECLARE
  vAlertButton NUMBER;
BEGIN  -- Initial processing
  vAlertButton := Show_Alert('Caution');
  IF vAlertButton = ALERT_BUTTON1 THEN
    NULL; -- Continue processing
  END IF;  -- do nothing for button 2
END;
```

用少量代码，也可以创建可重用的警报对象和警报，来显示引用专门的值或对象。例如，只用一个OK按钮创建单个指示性的警报。通过在运行时设置信息文本，可以再利用这个对象，而不必为每个信息创建一个单独的警报。可以用以下代码在库中建立一个简单的程序。

```
PROCEDURE Show_Info_Alert(pText IN VARCHAR2) IS
  vAlertID ALERT := Find_Alert('GenericInfoAlert');
  vDummy NUMBER;
BEGIN  Set_Alert_Property(vAlertID, ALERT_MESSAGE_TEXT, pText);
  vDummy := Show_Alert(vAlertID);
END;
```

然后以信息行的方式传递文本参数到这个程序调用中，用传递的信息显示 GenericInfoAlert警报。这个程序有意忽略从Show_Alert返回的代码。

还可以建立更复杂的可重用的函数，使信息接受参数，通过并置参数创立信息文本，显示动态信息，并返回适当的按钮编号。为了在其他的模块中再使用，应该把这个函数放入库内。

```
FUNCTION Show_Info_Alert(pText1 IN VARCHAR2, pText2 IN VARCHAR2)
      RETURN NUMBER IS
  vAlertID ALERT := Find_Alert('GenericInfoAlert');
  vMessage VARCHAR2(200);
BEGIN
  vMessage := 'The '||pText1||' is '||pText2||'.';
  Set_Alert_Property(vAlertID, ALERT_MESSAGE_TEXT, vMessage);
  RETURN Show_Alert(vAlertID);
END;
```

6.2 数据块

Oracle Developer提供了几个与块和块的项有关的特殊特性。用这些特点可以显著地改善表单的使用性能。如果使用列表，那么对不同的用途，可以用几种不同的方法构造它们。利用单选按钮可以容易地创建互不相容的选择项，利用计算域使总和数的自动显示和表单中的其他计算工作变得非常容易。使用正确的日期格式掩码，可以大大减少数据库中与日期和时间有关的错误。最后，可以利用程序的封装与 Data Block Wizard配合，把数据块建立在存贮的程序的基础上而不是表的基础上，从而增加一个面向对象和对系统隐藏数据的方法。

6.2.1 列表

列表是一个表单的项目。它在独立的位置显示文本项的列表。理想的情况是通过查看选择列表并选择一项让用户在几个互不相容的选项中选择。已经见过两种列表 —— 在第4章中

China-pub.com
下载

WorkHasSkills表单的下拉列表，和Designer 中见过的用来显示选择的值的列表 (LOV)。

有四种类型的列表，其中三个是列表项：

- Poplist ：文本项下拉列表。通过靠近列表框的小箭头激活它，使列表在项目附近的窗口上弹出。
- Tlist：在滚动列表上显示所有文本项的列表框，像一个单独的画布，但在表单的单个域内。
- Combo box （组合框)：下拉列表和文本项的组合。可以拉出文本项的列表，但也可以输入新值。
- List of values(LOV)(值的列表)：一种对话框。它显示正文项并提供用于列表中搜索的模型匹配能力。

这个对话框返回所选择的单个列表项，并把它拷贝到另外的块项。在 Form Builder中，这是一个完全独立的对象，不是列表项，而且是用每个项的专门的一组属性把它连接到项上。LOV提供在一个值的长列表中寻找一个项时使用模型匹配的能力。有关利用 LOV向导为项创建LOV的辅导，请参看第4章的内容。

所有这些列表都以某种方法显示文本项的列表，但这些字串不需要项目的值。列表把你在屏幕上看到的标签之类的东西转换为项上的一个实际值，即 Forms存贮在数据库内的值。由于标签和值之间的间接关系，可以用文本显示那些在域中使用其他数据类型的内容。也就是说，列表给了这样的一种方法：在表单中显示格式精美的有意义的正文，但在数据库中仍然存贮没有意义的、难以使用的代码。

例如，可以利用一个比较旧的、以数字形式存储不同选择代码的数据库。数字 1代表红的，2代表绿的， 3代表紫的，等等。列表项给用户显示的是文本字串“绿的”，而把数字2放入数据库中，在从数据库查询记录时，用“绿的”去置换 2，而不显示“绿的”这个字串。

1. 有效的列表

列表是用有效的方法显示信息的有力工具，有几种构造列表的方法使它们更实用。

我建议把所显示的字符当作数据库中的值使用。代码一般在非常需要磁盘空间并且程序员只需要读取数字时使用。现在，大多数的程序开发者学会了读正文。使用代码的理由是存在的，但是对大部分情况来说，是因为必须要适应“以前拥有”的数据库，那些数据库的设计没有使用现代的设计方法，没有对有数十亿实例记录值的非常大的数据库的空间进行优化。如果可以选择的话，保证列表的标签和数据库的值相同，可以使它变简单。

提示 在查询一组记录时，由于某种原因，数据库服务器返回的记录违犯触发器或表单中某些其他条件的要求，那么Oracle Developer会丢掉这个记录，但不告诉你。为什么我在这里提出这个问题呢？这些要求的条件之一是列表项的值要在代码列表中。如果从数据库中查询一个记录，有一个不在列表中，那么表单就不显示这个记录，而且不声不响地把它丢掉。表单假设正在使用的数据库的值和装入应用程序的确认条件是一致的。如果这个假设不成立，可能会引起混乱。应该确保数据库和在应用程序中建立的业务规则的设置的一致性。

列表项中Other Values属性存在的主要原因是因为有一种可能性，即：数据在数据库中有值，但在应用程序列表中却没有值。这种可能性不包括 LOV对话框，因为它不是列表项而完全是单独的一种对象。列表项的这个属性缺省设置是 NULL。如果不把它设置为某个值，那么

China-pub.com 下载

Forms Runtime会丢掉那些在列表上没有值的记录。如果把它设置为某个值，那么 Forms Runtime将显示这个记录，但是，显示的是 Other Values属性的值，而不是数据库的值。列表在Forms Runtime不显示对应于这个值的任何字符串。这个值可以是类似于“ Other ”那样的内容。必须把这个值加到列表中，给它一个适当的内容。

注意 不要求你提供Other Values的设置。例如，在检查与项对应的数据库列的约束的情况下没有“ other ”值从数据库中返回，可以设置Other Values为NULL，不会引起任何问题，但是如果不能保证返回的值在列表中，那么最好定义这个属性，不然表单的性能会相当混乱。

组合框为确认问题创建另一个机会。组合框总的思想是允许用户在域中自由输入文本，并且下拉一个提供选择值的列表。但结果是： Developer并不确认对列表所输入的正文。如果这是所希望的特性，很好。如果不是，那么将需要增加确认的编码。或解释违犯约束的数据库错误，或解释用户用某种方式输入的正文。也可以利用某些内部的管理列表项的子程序，把用户输入的项加到列表中去。这要求相当高级的编码。所以，最好是利用组合框作为建议性的列表，而不用去承担编制确认程序的工作负担。

2. 硬代码列表

硬代码列表(Hard-Coded List)是在Form Builder中完全规定的列表。用户用字串和相应的值填满一个对话框，这些值便成为用户的应用程序的一部分。这种方法的优点是简单和快捷，可以非常迅速地创建这些列表，而且在打开应用程序时这个列表是在存贮器中，所以运行得非常好。缺点是如果不重新编制应用程序就不能改变这些列表。对那些简单的不改变的列表可以使用硬代码列表：

创建硬代码列表的步骤：

1) 在块中创建列表项。

2) 显示这个项的Property Palette。

3) 设置项的Type Property为List。

4) 把Width属性设置为想要输入的最长的字串标签的长度。

5) 进入Property Palette的Function部分，把List Style属性设置为列表的三种类型之一：Poplist、TList或Combo Box。

6) 双击列表属性中的Elements。

将看到List Elements 对话框如图6-11所示。

7) 输入标签和相应的值。

8) 如果想要使值为 NULL，则让 Value的单元为空白。

9) 利用DOWN-ARROW(下箭头)键或鼠标移动到列表的新项处，利用 CTRL-K键查看可以在这个对话框中使用的功能键列表。利用 CTRL-<键来删除当前所选择的列表项 (不要忘记按SHIFT键以得到“ < ”而不是“ , ”)。

10) 完成后，单击对话框上的OK。

11) 如果希望能够处理那些不在列表中的值，

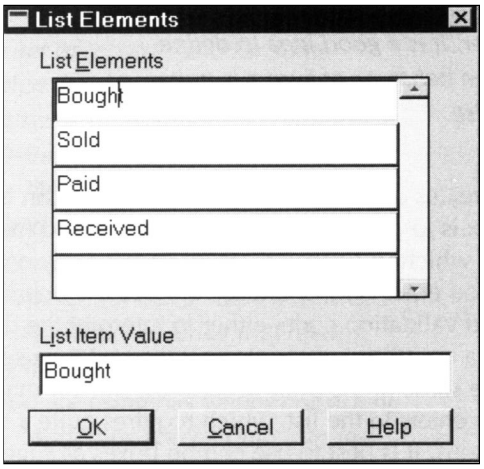


图6-11 List Elements对话框

China-pub.com
下载

那么在Other Values属性中输入一个值。

如果运行表单，当显示列表的内容时，将在列表中看到在对话框中输入的标签字串，可以选择列表中的任何项。

注意 有时，在列表中可能得到多余的空白行，这是由于删除了标签和值，又没有用CTRL-<删除列表元素而引起的。可以试试看。

3. 动态列表

动态列表是在运行时填入的列表。有两种方法创建动态列表。最普通的方法是根据数据库的表建立列表。列表由用户创建，是用SELECT语言填写的记录组对象得到列表的标签和值，然后根据记录组填充列表。另一种方法需要利用内部函数 Add_List_Element添加值。后面将较详细讲述第二种方法。首先让我们考虑根据表建立列表。

• 根据表建立列表

首先，如前面部分那样创建列表项。如果需要，在可以用 Other Values属性的List Element对话框内输入标签和值(参看前面的“有效的列表”部分)。

遗憾的是，这个值不能是NULL值。如果对单个硬代码列表的元素使用 NULL值，那么在准备从表创建新的动态列表时，就不能利用 Clear_List清除它，而会得到一个错误，且列表的加载失败。其原因是明显给出在版本1中出现的问题，即不希望在下拉列表中出现空白行。这个列表不允许删除NULL元素，也就不需要把它加进去产生不希望的空白行。这就意味着，必须为缺省值从数据库中选择一个值。例如，在Ledger表单的Person列表中使用“General Store”这个值作为缺省列表元素。当从表加载这个列表时，这个值将消失，但 Other Values属性中将仍然有这个值。于是这个值出现在从数据库加载的列表以及在硬代码的版本中。

这个要求的后果之一是，不能重新使用目标库中的标准列表 SmartClass。必须采用定义列表元素和Other Values属性的方法来规定列表项和它的数据库值。这也意味着如果选择的值在数据库中被改变了，将必须在硬代码的列表的 Elements属性中改变它。这是很遗憾的，但必须这样做。

一旦建立了 Other Values属性，就可以用通常的方法为表单创建一个记录组对象。在Designer中显示一个对话框，询问是根据值的列表还是用查询创建记录组。缺省设置是用查询，用单选按钮选择查询，并在文本框中输入一个查询。这个查询选择与标签和值对应的数据，并且按它们排序。如果值是一个数字或日期，那么利用 To_Chat函数把这个值转换成字符串。利用ORDER BY子句按照所希望在显示列表中呈现的次序进行排序。

例如，如果希望为Ledger表的Action列创建一个列表，那么应该使用以下的查询：

```
  SELECT DISTINCT Action, Action
    FROM Ledger
ORDER BY 1
```

注意 在ORDER BY子句中使用数字1可以按照标签排序而忽略存在有两个数据列使用相同名字的事实。

这个查询利用 Ledger表中的Action的唯一值产生一个标签和值均相同的记录组。利用GROUP BY子句得到这些唯一的值。 GROUP BY也按照标签将这些值排序，但是无论如何，加ORDER BY子句是一个很好的编码方法，使得在以后需要改变 GROUP BY时不致于混淆。

现在在表单中或在附加库中创建以下的程序。这对整个系统来说都是一个好的候选方法，

China-pub.com 下载

因为可以在任何表单中重用这个程序。

```
PROCEDURE Set_Up_List (pList in VARCHAR2, pGroup in VARCHAR2) IS
  vErrFlag NUMBER := 0;
  eListPopulationProblem EXCEPTION;
BEGIN
  vErrFlag := Populate_Group(pGroup);
  IF vErrFlag = 0 THEN
    Clear_List(pList);  -- remove anything already there
    Populate_list(pList, pGroup);
  ELSIF vErrFlag = 1403 THEN  -- no list elements found
    null;  -- do nothing and ignore the problem
  ELSE
    RAISE eListPopulationProblem;
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    Message('Exception:  Could not populate '||pList
            ||' with query in group '||pGroup||'.');
END;
```

注意 如果对同一个列表调用这个程序若干次，只需要清除这个列表。如果不清除这个列表，那么只是把已经不在表单上的值加到列表单上去。还要仔细检查从 Populate_Group返回的代码，因为返回的代码指出SQL查询中的某些问题。还要查看Clear_List属性、List Element属性和Other Value属性之间关系的注释。

现在，需要选择一个触发器来建立这个列表。可以在通过 When-New-Form-Instance触发器打开表单时建立这个列表，或者可以等待到准备显示这个列表时为止，也许在这个列表出现的块的When-New-Block-Instance触发器中。关于这一点，可以有相当大的创造性使得在加载表单时节省一些处理时间。现在在触发器中输入以下代码：

```
Set_Up_List('Action', 'LedgerActionGroup');
```

这段程序用它的内部查询填写LedgerActionGroup，然后，使用这个组来填充这个列表。

注意 也许想重新建立一个列表在应用程序中单独起作用。例如，在一个子表单中要修改出现在列表中的一组值，那么需要再一次运行建立列表的代码，重新设置列表，让它包含这些新的值。遗憾的是，这要求完成所有现有的使列表项出现在列表中的块的事务。做这项工作的一个好方法是在重新建立列表的代码中发送一个 Clear_Block或Clear_Form语句。

6.2.2 单选按钮

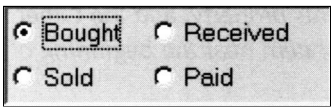


图6-12 单选按钮列表

通常，如果希望用户在一系列可能的值中作选择，那么应该提供一个供选择的列表。单选按钮提供了这样一种列表的方法。单选按钮组是一些无线电按钮的集合。无线电按钮是带有圆圈的小按钮(准确的格式取决于图形用户接口平台)，当用户单击其中一个按钮时，Oracle Developer离开当前的按钮而选择被选中的按钮。因此，每次只能选择一个按钮。无线电按钮这个名字出自收音机中选台按钮的特性，它们是按下一个按钮则弹起另一个按钮(或现代无线电的电子按钮)。在图6-12中，在每一种情况下只有一个选择是起作用的。

China-pub.com
下载

当有相对小数目的相互不相容的选择时，通常这些选择是两个或三个不随时间变化的选择，使用单选按钮。在上例情况中，有四种可能的动作类型是非常稳定的。

注意 单选按钮组结合应用程序的结构来构造对数据库的值的约束。因此，动态列表通常是更可取的接口。因为用户可以扩展它而不改变应用程序的代码。如果列表是静态的而不是动态的，或者，如果值实际上不随时间变化，那么单选按钮组在显示和容易使用等方面有某些优点。

单选按钮组是对应于数据库表中的列的块项，单选按钮对应于列的可能值。单选按钮在单个单选按钮组的项目下聚集在一起。当用户在按钮上单击时，这个按钮对象的 Value属性就变成了这个单选按钮组的值。

这个逻辑使这些对象所具有的一组属性互相有牵连，即影响所有按钮的任何属性都是按钮组的属性。比如缺省值与数据库有关的属性、导航的属性就是如此。只影响单个按钮的属性是按钮的属性，比如按钮值、按钮的标签和按钮的物理位置等。要在 Oracle Developer中设置那些为选择新记录而显示的初始化按钮，可以把单选按钮组的 Default Value属性设置为按钮的值(比如在这个例子中的“ Bought ”)。

注意 在块中创建记录之前，这个块的任何项的Default Value属性不起作用。即使是可以看见这个块，在导航到它上面以前，实际上也并没有在这个块上创建记录。在实际在块上创建新记录以前，缺省值并不出现。

为了要建立单独的按钮，以通常的方法利用 Create工具创建，然后改变Label属性和Radio Button Value属性，Label是Oracle Developer作为提示而显示的文本，Radio Button Value是当用户单击按钮时，Oracle Developer赋予单选按钮的实际值。

至于列表，如果用一个与任何单选按钮值不对应的值从数据库中取回一行，那么就会发生问题，Oracle Developer将悄悄地丢掉这个记录。单选按钮组的 Other Value属性工作起来就像有同样名字的列表项的属性一样，它让你为了显示的目的把和按钮不对应的值变换为一个按钮值。

根据用户界面设计的类型，通常用某种可视化方法区分单选按钮组比较好。前面的例子利用样板图形的长方形包围着四个单选按钮，把它们和画布上其他的块项分开。在这种情况下，长方形的Beveled属性设置为Lowered ，可以显现出三维的效果。

6.2.3 计算项

创建表单的一个共同任务是创建表示某个计算值的项。例如，可以把几个数据库的项组合成单独的连接起来的字符串，而这些项在基础表中不是一一对应的。在 Ledger表单中，可以用Quantity(数量)乘Rat(价格)的计算域来代替数据库的Amount (总计)域。另外一个普通的例子是当创建Total域时，在另一个块中计算记录总数。

在Oracle Developer以前的版本中，通过利用各种触发器来计算这些值，并通过把值赋予项来显示它们从而达到这些效果。通常是利用 When-Validate-Item和Post-Query触发器达到这些效果。例如，要计算 Amount (总数)，就把Base Table Item属性设置为False，通过改变项的类型使项成为显示项，然后对 Quantity (数量) 和Rate (价格) 创建 When-Validate-Item触发器，每一个触发器有这样的代码：

China-pub.com 下载

```
:Ledger.Amount := :Ledger.Quantity * :Ledger.Rate;
```

这个处理插入和更新了两个数据库的项。还应该用同样的代码在 Ledger块创建一个Post-Query触发器来处理从数据库返回的数据。

利用Calculation属性组，可以非常容易地创建计算域。通过设置项属性 Database Item为NO，创建一个不在基础表中的显示项。不创建触发器，而是设置 Calculation组的属性。把Calculation Mode (计算模式)设置为Formula (公式)产生计算特性，然后输入公式，就像前面见到的输入 PL/SOL代码那样：

```
:Ledger.Quantity * :Ledger.Rate
```

每当记录中发生某种变化时， Oracle Developer再计算这个域并显示新的值。

在Oracle Developer中进行总计统计是很容易的。在想要总计的块中创建总计项，必须把这个块的Advanced Database 组的Precompute Summaries (预先计算累加) 属性设置为Yes，这就告诉Oracle Developer，在做查询以前先计算总和。因为通常这个块会显示多个记录，然而总计时并不想对每个记录显示一个总计，所以把要显示的总计项的属性数目设置为 1(通常它为零，意味着项的数目和在块中显示的记录数目相同)。

注意 把Query All Records in Records group(查询记录组中所有的记录)设置为Yes，而不是把Precompute Summaries (预计计算总和)设置为Yes，可以达到同样的效果。与其分别计算总和，不如Oracle Developer查询所有的记录而不是在需要时提取它们。这就可以从存贮器中的记录来计算总和。对于有大的结果集的块，这可能会引起一些存贮器问题。预先计算也会慢下来，但不像Query All Records(查询所有的记录)那么严重。用域的性能测试进行实验，决定使用哪种方法。

现在，设置项的四种属性， Calculation Mode (计算模式)设置为Summary (求和)，决定了项的求和计算。把Summary Function设置为Sum，告诉Oracle Developer要添加记录的值。可以利用任何标准的总计功能，比如 Count、Average、Min、Max、Stddev或Variance。Summarized Block和Summarized Item下拉列表分别显示所有的块和所选择的块中的所有的项。例如，要创建一个Total (总计)项来总计Ledger块的Amount (数量)项，就把Summary Function (总计功能)设置为 Sum (求和)，对Ledger块求和 (Summarized Block)，而对 Amount (数目)项求和 (Summarized Hem)。Oracle Developer自动地在画布的独立的域中显示对块的总计，并处理对求和项所有的计算的变化，包括清除或除去记录。

6.2.4 日期项和时间项及格式掩码

和计算机打交道的大多数人现在必定认识到了“ 2000年 ”问题。当时钟在1999年的最后一天的夜里敲响时，所有两位数字的年域将变为“ 00 ”，而且根据媒体的说法，这个结果将是混沌的(Chaos)。这本书现在还是“ 温顺的 ” (Compliant)！在2000年1月1日以后你仍然可以读。

这个问题是讲得过分了，但是的确要注意日期域和时间域。为了确保没有 2000年的问题，应该把年号规定和显示为四位数字值。对于 Oracle Developer的日期项，用某种“ YYYY . ”似的变化转换成格式掩码。日期格式掩码可以处理成世界各地普通的日期格式，所以不必要书写大量的触发器或程序来格式化日期。

大多数日期和时间问题发生在进入另一个不完整的日期和时间场所时。处理时间的问题

China-pub.com
下载

可能相当严重，除非你记住了有关 Oracle Developer和Oracle时间的几个基本事实。

生活中最主要的事实是，时间和日期是 Siamese孪生子，它们不能够摆脱地连在一起。对 Oracle Developer的项和域， PL/SOL变量和Oracle的列，日期类型DATE是共同的，并包括有日期和时间两部分。大多数问题发生在企图单独处理日期和时间的时候。例如，有可能想用单独的可输入日期和时间的域呈现给用户，让他们设置或查看时间。如果不在 Pre-Insert、Pre-Update、Pre-Delete和Post-Query触发器中进行PL/SOL编码，利用To_Char和To_Date转换功能组合和分开日期元素，就不能这么做。为了避免不必要的压力，在 Oracle Developer中，日期项使用格式掩码，例如，“ MM/DD/YYYY ”“ HH：MI ”，而不要为了“ 可用性 ”而将它们分开。

注意 Oracle Developer Forms为了和前面版本的表单兼容，也有DATETIME和TIME数据类型。不要使用这些数据类型，因为转换为Oracle DATE可能产生意外的结果。

使用Oracle Developer的格式掩码并不影响实际日期和时间的正确性，它会总是保持相同。在屏幕上看到的并不是进入数据库得到的，因此，对于这个问题没有任何事情要做。例如，在一个表单的日期项中没有看到时间，这并不意味着没有时间或时间是 00：00，虽然这可能是由缺省得到的结果。可以用 PL/SOL代码设置特定的时间：

```
:Ledger.ActionDate := To_Date(To_Char(:Ledger.ActionDate,
                                      'MM/DD/YYYY')||' 12:00',
                              'MM/DD/YYYY HH:MI');
```

这个代码指定，通过 Ledger.ActionDate插入的所有项的日期的缺省值为中午 12点，而不是半夜的12点。

表6-3显示了和年份有关的Oracle Developer格式掩码，可以组合成日期-时间格式掩码。

表6-4显示了可以在Oracle Developer格式掩码中使用的和时间有关的各种元素。

表6-3 与年份有关的格式掩码元素

元 素	说 明
A.D.	对B.C.日期显示“ B.C. ”，对A.D.日期显示“ A.D. ”
AD.	对B.C.日期显示“ BC ”，对A.D.日期显示“ AD ”。
B.C	对B.C.日期显示“ B.C. ”，对A.D.日期显示“ A.D. ”
BC	对B.C.日期显示“ BC ”，对A.D.日期显示“ A.D. ”
RR	两位数字的年份，世纪为缺省最好使用四位数字而不使用两位数字
RRRR	四位数字的年份，与YYYY相同
SYYYY	带有数字的年份和一个负号的B.C.日期
Y	一位数字的年份(不推荐)
Y，YYY	四位数字和一个千位分隔逗号的年份
YY	两位数字的年份(不推荐)
YYY	三位数字的年份(不推荐)
YYYY	四位数字的年份(推荐)

表6-4 与小时有关的格式掩码元素

元 素	说 明
A.M.	对ante-meridian时间为“ A.M. ”，对post-meridian时间为“ PM. ”
AM	对ante-meridian时间为“ AM ”，对post-meridian时间为“ PM ”
HH	一天中的小时用12小时格式(1-12)
HH12	一天中的小时用12小时格式(1-12)

China-pub.com 下载

(续)

元　素	说　明
HH24	一天中的小时用24小时格式(0-23)
MI	小时的分钟(0~59)
P.M.	对ante-meridian时间为“ A.M. ”，对Post-meridian时间为“ P.M. ”
PM	对ante-meridian时间为“ AM ”，对Post-meridian时间为“ PM ”
SS	分钟的秒(0~59)。
SSSSS	自午夜后的秒数(0~86399)

表6-5展示了和日期及月份有关的元素，可以组合格式掩码。

表6-5　与日子和月份有关的格式表征码元素

元　素	说　明
D	一周的数字日期(1~7，1为星期天，除非当前的NLS语言把它定义为别的东西，比如星期一)
DAY	固定长度，九个字符的日子名(“ SUNDAY ”、“ MONDAY ”等等)
DD	一个月中的日期(1~31)
DDD	一年中的日期(1　366)
DY	三个字母的日期名(SUN、MON、TUE等等)
J	自公元前4712年1月1日以来的天数(“ Julian ”)只在专业的排序应用程序中使用，Oracle公司反对使用这种格式，所以不要使用它
MM	数字格式的月份(1~12)
MON	三个字母的月份名(JAN、FEB，等等)
MONTH	固定长度，九个字符的月份名(“ JANUARY ”、“ FEBRURY ”等等)

可以把大多数标点符号放在格式掩码的位置，而 Oracle Developer在显示字符串中重新创建它。可以把在双引号内的任何文本串嵌入掩码来显示它。例如，掩码“　MONTH ”“ DD ”，“ YYYY ”显示字符串“ APRIL　18，1999 ”。

注意固定长度的MONTH(月份)，表征码元素使月份名字中产生了额外的空格，可以利用在字串行前加 FM，消除前面的空格“ FMMONTH ”“ DD ”，“ YYYY ”，则字符串显示为：“ APRIL 18，1999 ”。可以在格式元素中规定字体，根据自己的喜好使用“　MONTH ”或“ Month ”。

定界符是在日期中分开年、月、日的标点符号标签，通常是破折号或斜线，例如 18-APR-1999或4/18/1999。当把它们作为格式掩码的一部分时，用户可能会输入不同的标点符号 (例如4.18.1999)，掩码则会适当地转换它们。为了强制使用特定的标点符号和强制用户提供日期或时间中的数字和字符，可以使用 FX前缀。例如格式掩码FXDD-MON-YYYY HH：MM就是强制用户用标准的格式输入：01-JAN-1999 12：00，包括前面的0和破折号。

可以同时使用FX和FM前缀，使FX只用于限定符。例如，FMFXDD-MON-YYYY将允许用户输入1-JAN-1999，而不能输入1.JAN.1999。

6.2.5　过程的封装

EXECUTE特权给出一个有趣的方法可以给对象授予广泛的特权：程序封装。任意接近用户、角色和特权的方法的缺限是基于服务器的，这些特权的应用忽略了用来会话的机构。就是说，当通过应用程序注册时，可以得到修改数据的特权，就像通过 SQL*PLus或另外一个程序注册时得到的特权一样。如果某些或全部的确认进程只以 Oracle Developer的代码存在，那

China-pub.com
下载

么由于缺少确认，在通过其他的程序时可能会引起数据库问题。如果依赖于应用程序层的安全性，那么问题可能更严重，因为服务器并不实施这种安全性测量。

如果真想要在高度安全环境下限制用户的能力，可以把所允许的特性封装在存贮的程序中，然后在程序中给角色或用户授予 EXECUTE(执行)特权。用户并不需要基础对象的特权来执行这个程序，只需要创建这个程序本身的那些特权。这样，通过运行程序的应用程序注册的用户，就可以做他需要做的事情。但是通过 SQL*PLus注册的用户不能用 SQL直接做这些事情(当然，不是运行带有隐含限定的同一个程序)。

注意 这个讨论也适用于触发器以及程序。

Oracle Developer有能力使用存贮在作为数据源的数据块中的程序。首先，考虑数据块和作为数据库对象类的基础表，需要作什么能够与这些作为类的对象相联系呢？

- 选择(Select)：选择一些数目的数据库对象并作为数据块的记录显示它们。
- 插入(Insert)：用数据块的记录插入数据库对象。
- 更新(Update)：用数据块中被修改的记录去改变数据库对象。
- 删除(Delete)：从在数据块中选择的记录中删除数据库对象。
- 锁定(Lock)：锁定与块记录对应的数据库对象。

这五件最基本的事情就是 Oracle Developer在数据库中做的大部分的事情。如果为每一件事情都建立一个存贮的程序，然后把它们和数据块连接，那么就可以享受到对 Oracle Developer应用程序的所有程序封装的优点。任何人用这些程序上的 EXECUTE(执行)特权来运行应用程序，都将会看到应用程序的运行就像是使用标准的数据块一样。而没有这些特权的任何人将不能在应用程序中做任何事情。还有，如果把广泛的表查找作为数据库确认的顺序的一部分的话，那么把所有这些移到服务器作为存贮程序的一部分，可以改善性能。

要用这种方法使用程序封装，必须要为作为 Oracle Developer应用程序的数据块基础的每一个表，建立一个专门的程序包。例如，对 Skill表，可以定义这个程序包，将它连接到拥有 Skill的用户上：

```
CREATE OR REPLACE PACKAGE SkillPackage AS
    -- A record type with all the columns in the Skill table
  TYPE tSkill IS RECORD (
    rSkill Skill.Skill%TYPE,
    rDescription Skill.Description%TYPE);
  -- A record type with the primary key column in the Skill table
TYPE tSkillKey IS RECORD (rSkill Skill.Skill%TYPE);
  -- A ref cursor for a set of Skill rows
TYPE tSkillCursor IS REF CURSOR RETURN tSkill;
  -- A table type for a table of Skills
TYPE tSkillTable IS TABLE OF tSkill INDEX BY BINARY_INTEGER;
  -- A table type for a table of primary keys;
TYPE tSkillKeyTable IS TABLE OF tSkillKey INDEX BY BINARY_INTEGER;
  -- An exception for the row-locked error (Oracle error -54)
eRowLocked EXCEPTION;
PRAGMA EXCEPTION_INIT(eRowLocked, -54);
  -- Two query procedures, one by cursor and one by table
PROCEDURE SelectCursor (pSkillCursor IN OUT tSkillCursor,
                        pDescription IN VARCHAR2);
PROCEDURE SelectRows (pSkillTable IN OUT tSkillTable,
                      pDescription IN VARCHAR2);
```

China-pub.com 下载

```
    -- Data manipulation procedures
  PROCEDURE InsertRows (pSkillTable IN tSkillTable);
  PROCEDURE UpdateRows (pSkillTable IN tSkillTable);
  PROCEDURE DeleteRows (pKeyTable IN tSkillKeyTable);
  PROCEDURE LockRows (pKeyTable IN tSkillKeyTable);
END SkillPackage;
```

这个程序包是抽象数据类型的一个例子——由一组特性伴随的一组数据结构，这些特性对这组数据结构进行操作。它类似面向对象的类，但它没有继承性。

注意 要了解PL/SQL的意思，需要参考与PL/SQL有关的第11章和第12章。这些章节讨论在程序包技术条件中看到的各种元素。第 12章还包含关于结构的细节和与 Oracle Developer一起的PL/SQL程序包的使用。

程序体可以包含所需要的任何的 PL/SQL代码，虽然它们通常必须在 Oracle Developer期望的程序方向上运行。

```
CREATE OR REPLACE PACKAGE BODY SkillPackage AS
    --
    -- Query by reference cursor
  PROCEDURE SelectCursor (pSkillCursor IN OUT tSkillCursor,
                          pDescription IN VARCHAR2) IS
    vDescription VARCHAR2(80) := pDescription;
BEGIN
  IF vDescription IS NULL THEN  -- select all skills for NULL
    vDescription := '%';
  END IF;
  OPEN pSkillCursor FOR
    SELECT Skill, Description
      FROM Skill
     WHERE Description LIKE vDescription
     ORDER BY Skill;
END SelectCursor;
  --
  -- Query into table of records
PROCEDURE SelectRows (pSkillTable IN OUT tSkillTable,
                      pDescription IN VARCHAR2) IS
  vIndex NUMBER := 1;
  vDescription VARCHAR2(80) := Nvl(pDescription, '%');
  CURSOR cSkill IS
    SELECT Skill, Description
      FROM Skill
     WHERE Description LIKE pDescription
     ORDER BY Skill;
BEGIN
  OPEN cSkill;
  LOOP
    FETCH cSkill INTO pSkillTable(vIndex).rSkill,
                      pSkillTable(vIndex).rDescription;
    EXIT WHEN cSkill%NOTFOUND;
    vIndex := vIndex + 1;
  END LOOP;
END SelectRows;
  --
  -- Insert rows from a record table into database table
PROCEDURE InsertRows (pSkillTable IN tSkillTable) IS
  vIndex NUMBER := 1;
```

China-pub.com
下载


```
  vCount NUMBER := pSkillTable.Count;
BEGIN
  FOR vIndex IN 1..vCount LOOP
    INSERT INTO Skill (Skill, Description)
    VALUES (pSkillTable(vIndex).rSkill,
            pSkillTable(vIndex).rDescription);
  END LOOP;
END InsertRows;
  --
  -- Update rows from a record table into database table
PROCEDURE UpdateRows (pSkillTable IN tSkillTable) IS
  vIndex NUMBER;
  vCount NUMBER := pSkillTable.Count;
BEGIN
    FOR vIndex IN 1..vCount LOOP
      UPDATE Skill
         SET Description = pSkillTable(vIndex).rDescription
       WHERE Skill = pSkillTable(vIndex).rSkill;
    END LOOP;
  END UpdateRows;
    --
    -- Delete rows from a table of keys
  PROCEDURE DeleteRows (pKeyTable IN tSkillKeyTable) IS
    vIndex NUMBER := 1;
    vCount NUMBER := pKeyTable.Count;
  BEGIN
    FOR vIndex IN 1..vCount LOOP
      DELETE FROM Skill
            WHERE Skill = pKeyTable(vIndex).rSkill;
    END LOOP;
  END DeleteRows;
    --
    -- Lock rows from a table of keys
  PROCEDURE LockRows (pKeyTable IN tSkillKeyTable) IS
    vIndex NUMBER := 1;
    vCount NUMBER := pKeyTable.Count;
    vDummy VARCHAR2(1);
  BEGIN
    FOR vIndex IN 1..vCount LOOP
      SELECT 'X'
        INTO vDummy
        FROM Skill
       WHERE Skill = pKeyTable(vIndex).rSkill
      FOR UPDATE NOWAIT;
    END LOOP;
  END LockRows;
END SkillPackage;
```

注意 这种逻辑的有效使用，使得不能使用标准的Oracle Developer的query-by-example机制。你必须通过把变量传递到选择程序，自己建立查询。在刚给出的例子中，两个选择程序进行了说明，说明哪一个是可以查询的唯一列。程序体中的WHERE子句假设你想在这个值的输入中做一个LIKE表达式。你可以做得比这复杂得多，但这种技术超出了这本书的范围。

1) 编译作为程序包拥有者注册的程序包 (例如Talbot)。

China-pub.com 下载

2) 给需要调用这个程序包的程序的每一个角色授予在这个程序包上的 EXECUTE特权(例如HRSkillRole)。

```
GRANT EXECUTE ON SkillPackage TO HRSkillRole;
```

3) 作为DBA用户或作为有CREATE PUBLIC SYNONYM特权的用户注册，并为这个程序创建一个公共的同义词：

```
CREATE PUBLIC SYNONYM SkillPackage FOR Talbot.SkillPackage;
```

注意 必须为能够通过应用程序访问包装程序的用户授予EXECUTE特权和创建公共的同义词，除非你把拥有者的用户名编码作为程序名的一部分，例如像 TALBOT.SKILL PACKAGE、SELECURSOR。通常，这并不被认为是好的编码方法，因为它把应用程序编码和特定用户连接起来了。

现在准备创建数据块。

4) 从Tools | Data Block Wizard中启动Data Block Wizard(见图6-13)。

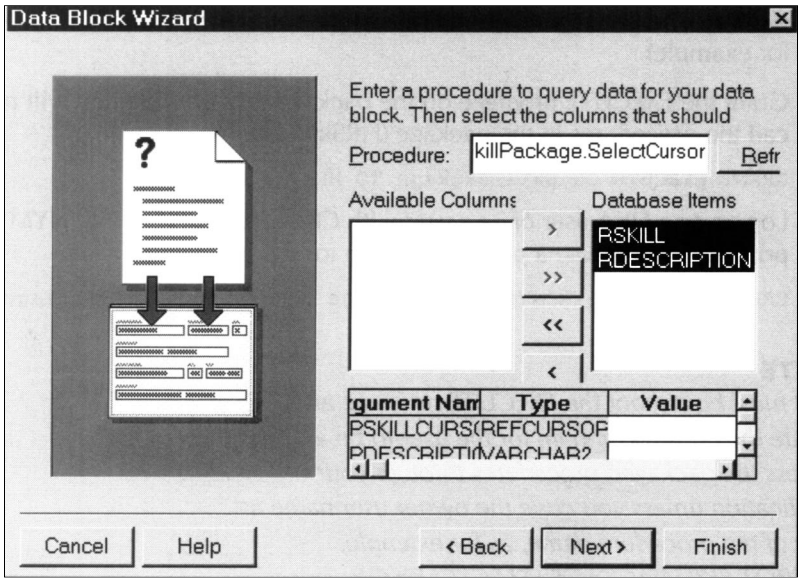


图6-13 Data Block Wizard对话框

5) 在第一个对话框中，选择Stored Procedure单选按钮，而不要选择Table按钮。这就告诉Wizard是程序的信息，而不是标准的表和列的信息。

6) 单击Next按钮，开始定义块。

7) 下一个对话框询问查询程序，输入程序包名和游标程序名。例如“ SKILLPACKAGE. SELECTCURSOR ”，遗憾的是这个版本不让你为了这个信息而浏览数据词典。

8) 在Refresh(刷新)按钮上单击，显示类型列的名字 (RSKILL和RDESCRIPTION)。

9) 用>或>>按钮来调整数据块所需要的列。注意，位于对话框底部的变量列表显示的是程序的变量。不要改变这些变量。

注意 确信你把程序包的名字加到了程序名字上，否则Data Block Wizard将找不到这个程序，在插图的对话框中程序包的名字滚动到了项域的左边。

China-pub.com
下载

10) 单击Next，移到下一个对话框，提示插入程序。

11) 输入程序包名字和插入的程序名：

“ SKILLPACKAGE.INSERTROWS ”。

12) 在Refresh按钮上再次单击，显示类型列和程序变量，虽然，在这里没有任何它们可以做的事情(见图6-14)。

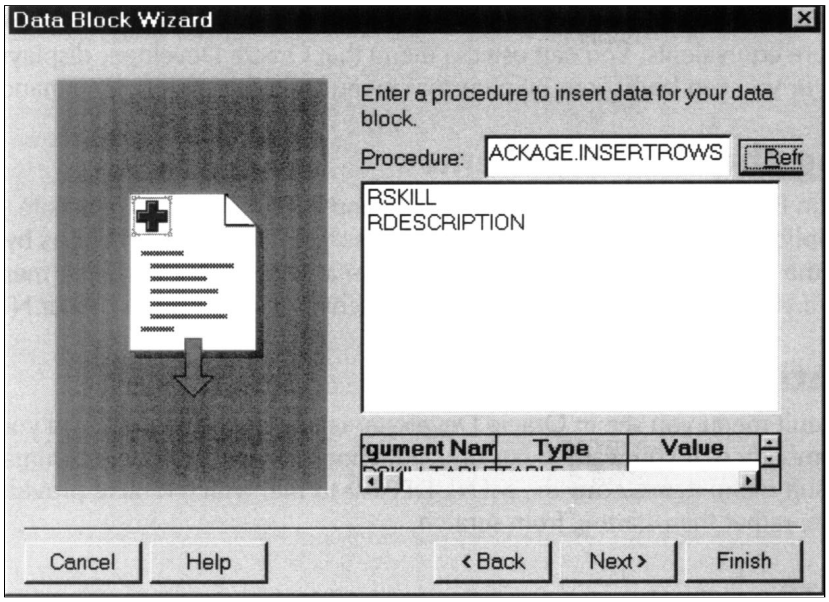


图6-14 单击Refresh，显示类型列和程序变量

13) 在下面的三个对话框中重复更新、删除和锁定这三个步骤 (分别是UPDATEROWS，DELETEROWS和LOCKROWS)。如果数据块结构使得有可能的话，会看到出现标准的主 -从关系定义对话框。这个窗口让你在新的块和先前定义的其他块之间定义任何主 -从关系。

14) 像往常一样，现在可以前进到Layout Wizard中。

利用程序封装，可以有限制地访问封装在 PL/SQL程序中的明确定义的动作特性，不容许广泛的SQL访问对象。

6.3 菜单

Form Builder的菜单系统让你执行运行时系统的不同的功能，比如 Next_Record，Enter_Query或Previous_Item。这些命令的大多数直接与按键对应，而且大多数有等效的内部程序，可以利用Oracle Developer显示缺省的菜单，或者可以建立自己定制的菜单以增加或改变命令。

6.3.1 利用缺省菜单

Form Builder系统有一个内部的命令菜单，在执行表单应用程序时，如果不定义菜单，那么系统显示缺省菜单，如图 6-15中的菜单。如果是在MDI环境下，比如在Windows 95或NT下运行，表单还显示缺省的菜单工具条。

China-pub.com 下载

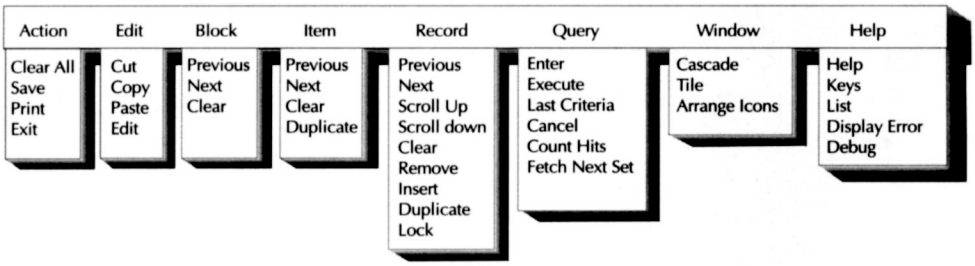


图6-15 Forms Runtime缺省的菜单结构

6.3.2 创建定制菜单

在Oracle Developer中看见的缺省菜单是在代码中生成的，而且不能修改它。如果想加一些命令到缺省的菜单项上来改制一个菜单，那么就拷贝 MENUDEF.MMB文件。Oracle Developer提供它作为样板，而不是从暂存区开始。

注意 提供自己菜单模块的一种结果是菜单工具条消失。像前面的“创建工具条”部分讨论的那样，这个工具条提供一些标准的操作，比如Enter Query和Next Record等。你不能通过定制的菜单来显示这个缺省的工具条。必须利用菜单的 Horizontal Menu Toolbar属性或Vertical Menu Toolbar属性的Visible 值来建立自己的工具条。

1. 创建菜单、项和子菜单

在菜单模块中有两种对象：

- 菜单：主菜单、个别的菜单(比如Action或Record)和子菜单(从菜单项弹出的分级菜单)。
- 菜单项：在每个个别菜单中或子菜单中的项，对应于 PL/SQL代码或命令文本，个别菜单或子菜单拥有菜单上的项。

每个菜单和菜单项都有一个名字，用这个名字可以在菜单模块中唯一地识别它。另外，菜单项有一个标签，在显示菜单时，字符串出现在菜单上。

在创建菜单结构时，应该遵循五条原则。

第一，任何个别菜单或子菜单不应有多于九个的菜单项。如果菜单比较短，则比较容易使用。如果需要，可以利用子菜单把几个菜单项收集入组。

第二，如果要使用子菜单，努力争取只在主菜单的个别菜单的下一层上使用它们。嵌套子菜单的层次越深，找命令就越困难。例如在 Query菜单中用Report子菜单，它有两个菜单项，Ledger Summary和Mailing Labels。这就意味着系统中菜单级不大于三，用户可以容易地找到这些菜单命令(见图6-16)。

第三，不要考虑根据应用程序当前状态变化的动态菜单。这样不会将用户弄得莫名其妙。菜单应该是稳定和不变化的。

第四，为了帮助用户避免错误，在应用程序当前状态下把不使用的菜单项置为禁止状态(变成灰色)。例如，如果显示一个特殊的无模式窗口，这个窗口使用一些命令，但不使用其他的东西，那么禁止不用于它的或不适用整个应用程序的菜单项。可以采用 Disable_Item内部子程序来做这项工作，这个子程序要给定菜单名和项名：

```
Disable_Item('Block', 'Previous');
```

这个例子禁止Block | Previous菜单项。要允许这个项，使用 Enable_Item，采用同样的参

China-pub.com
下载

数变量。

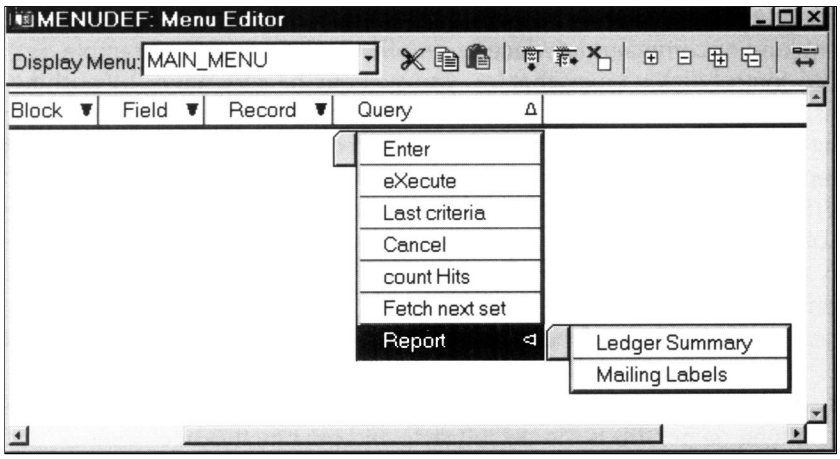


图6-16 嵌套子菜单

注意 通过使用Set_Menu_Item_Property内部子程序或通过分别在 Enable_Item和Disable_Item内部子程序中使用菜单名字MAIN_MENU，可以对主菜单上的项进行允许和禁止。

如果想显示菜单工具条，那么菜单项的次序就确定了工具条上图标的次序。因此，必须平衡菜单的透视次序和工具条的透视次序。另一方面，可以显示你自己的工具条，把它作为应用程序中的单独的画布。

如果有某些理由使你不能坚持这些原则，那么应该考虑用不同种类的命令接口来简化菜单。例如，建立一组控制对话框，通过按钮来执行命令，或建立一些对话框，采用输入选择作为选项而不是提供每一条命令作为菜单上的单个命令。不用列出作为菜单项的所有报表，可以有单个的菜单项，如Query | Run Report，有一个对话框用一系列单选按钮或复选框来选择运行报表。甚至可以从表示命令结构的图形显示中得到好处。发挥创造性，使你的应用程序更容易使用。

有三种类型的菜单，本书只详细讨论其中的一种：

- 下拉菜单：采用在单独的菜单名字上单击方式的菜单，这是缺省的类型，本书使用这种菜单。
- 全屏幕(Full-screen)菜单：占据了字符模式的整个屏幕的菜单，通过选择项与显示新屏幕来定位命令。这些菜单只在字符模式和块模式的应用程序中工作。
- 菜单条类型(Bar-style)菜单：这是这样一种菜单，当在它的上面单击时，用单独的菜单项代替主菜单。这种菜单只占据菜单条的空间，而且决不扩展到应用程序的内容区域，这种类型对于字符模式的应用程序比GUI更好。

如果想知道更多的有关全屏幕或工具条类型菜单的信息，请参看联机表单资料。

在菜单可以使用以前，必须把菜单文件编译为 MMX文件。在试图运行使用这个文件的任何表单之前，利用Form Designer(表单设计程序)中的File | Administration | Compile File来产生MMX文件。在进行某些修改以后，还必须重新编译这个 MMX文件，因为在运行表单时，不会自动做这些事情。

China-pub.com 下载

注意 对开发者来说，在运行表单之前忘记编译菜单是非常普遍的。每当在菜单上工作时，应该养成习惯，利用与File | Administration | Compile或Compile File对应的按钮，比如Windows中的CTRL-T自动产生表单。

2. 编辑菜单

Menu Editor让你直接看到正在建立的菜单系统。必须了解它的几个特性，才能充分编辑菜单和菜单项。在每个菜单左上角的小标签是选择标签。用在这些标签上单击的方式来选择菜单对象，也可以通过单击标签并把它拖到新的位置来移动菜单。 CTRL-dragging是拷贝菜单而不是移动它。在任何菜单项上单击即选择了这个项。像在表单中那样通过选择 Tools | property palette项或通过在Object Navigator上的对象的图标上双击，就可以看到在对象的Property Palette上的任何对象的属性。在 Menu Editor的菜单上双击，则可以编辑它的标签。可以用Delete 工具删除一个被选择的对象。

编辑器工具条上的两个专门的工具，可以用它添加菜单项和子菜单。用 Create Down工具可以在所选择的菜单项下加一个菜单项到菜单中。用 Create Right工具可以在选择的菜单项右边创建一个子菜单。这些工具对应于 Menu | Create Down和Menu | Create Right菜单项。

可以用通常的方法改变Object Navigator或Property Palette中的菜单项的名字。改变项的名字并不改变项的标签，反之亦然。如果改变名字，在编辑器看来，菜单是同样的，因为并没有改变标签。在项上双击或在 Property Palette中用Label属性来改变标签。

注意 界面设计原则经常说，要在标签后面放一个省略符号(三个圆点)：如“Options... ”。像大多数这些原则指出的那样去做，但是许多用户界面的设计却失败了。只有当显示一个对话框要求更多的信息执行命令时才应该这么做。例如对子菜单或对通过菜单项调用文档窗口就不用这么做。

3. 编制菜单项代码

用菜单项的Command Type(命令类型)属性来指定那些通过选择项目来执行的命令的种类。参看表6-6。

表6-6 菜单项命令类型属性值

值	说 明
Null	不做任何事情，可以使用这个值创建分隔符项，或使一个项为空，直到准备要它某些事情为止
Menu	调用子菜单，命令文本是子菜单的名字
PL/SQL	执行命令文本中的一块PL/SQL代码，这个代码不能直接引用表单模块的变量，而必须使用Name-In和Copy间接引用它们
Plus、Current Forms、Macro	不要使用这些兼容的选择，使用Host或Run_Product来PL/SQL内部程序运行SQL*Plus、表单，或操作系统命令

利用在圆点分开语法menu_name.item_name中的菜单名和项名来引用 PL/SQL中的菜单项，可以利用这个语法把命令连接到菜单项、初始化项、改变启动代码，或使能 /禁止菜单项，利用语法main_menu.item_mane来引用主菜单上的项，比如Block或Query。

要创建编码，在菜单项的属性设置板的 Command Text属性上双击，并把编码输入到PL/SQL编辑器中，编辑它然后关闭编辑器。

注意 如果要执行一个简单、标准的与某种按钮对应的命令，那么使用 DO_Key内部

China-pub.com
下载

程序：

```
Do_Key('COMMIT_FORM');
```

这个PL/SQL代码执行Save项，利用DO_Key联合命令键和菜单项，使你不必重复工作。

如果要使用Oracle Developer Server，那么可以利用内部子程序Web.show_Document来运行报表并在 Web浏览器中显示它。如果使用一个大的应用程序，那么可以用内部程序Run_Product运行Oracle产品。例如Ledger Summary菜单项运行Ledger Summary报表，这个菜单项的代码看起来像这样：

```
Run_Product(REPORTS, 'ledger', SYNCHRONOUS, RUNTIME, FILESYSTEM,
            NULL, NULL);
```

这个命令用文件系统中的 Ledger.RDF文件运行 Runform程序。它立即运行这个报表(RUNTIME)而不是以批处理方式，而且它同步运行这个报表，这意味着必须等待报表完成打印后，才能继续应用程序的工作，这里没有参数也没有图形显示。

注意 结尾的两个NULL值是被要求的。它们表示这个特殊的Run_Product调用没有使用变量。没有这两个NULL值，PL/SQL将不认识这个调用，因为变量的数目无效。

可以利用Run_Report_Object内部子程序更直接地运行报表。首先，必须在表单中定义一个报表对象，在你的表单下找出报表的标题，用通常的方法单击 Create工具。出现New Report对话框，它让你从Scratch或通过从文件中读入一个报表模块来创建一个新报表。第 7章详细介绍利用Report Editor在表单中创建报表。一旦有了报表对象，就可以利用 Property Palette给它取一个名字并设置Dev/2000/Integration属性组的Execution Mode属性和Communication Mode属性(例如同步和运行时)。还可以为报表设置系统参数，比如 Destination Type、Name和Format。

一旦建立了报表对象，重新产生表单，现在可以在调用 Run_Report_Object内部程序中通过名字来引用报表对象了。例如如果利用 Ledger.RDF报表文件创建了Ledger报表对象，那么把这个代码放在Query/Ledger Summary菜单项的PL/SQL块中 ：

```
DECLARE
  vReturn VARCHAR2(100);
BEGIN
  vReturn := Run_Report_Object('Ledger');
  IF vReturn IS NOT NULL THEN
    Message('Ledger Summary Report: '||vReturn);
    Synchronize;
  END IF;
END;
```

要引用附加到菜单上的表单的项值，可使用 Name_In内部函数。例如，要使WORKER块的NAME项的值与一个特定的名字比较，应该使用以下代码：

```
IF Name_In('WORKER.NAME') = 'Adah Talbot' THEN
```

类似地，用Copy内部程序指定一个值：

```
Copy('Adah Talbot', 'WORKER.NAME');
```

如果PL/SQL的复杂程度适中，那么最好把这个代码放入和菜单模块结合的程序库的单独的程序单元中。例如刚才给出的运行报表的代码可以是一个要求报表的名字的程序，如以下所示：

China-pub.com 下载

```
PROCEDURE Run_Form_Report(pReport IN VARCHAR2) IS
  vReturn VARCHAR2(100) := Run_Report_Object(pReport);
BEGIN
  IF vReturn IS NOT NULL THEN
    Message(pReport||' Report: '||vReturn);
    Synchronize;
  END IF;
END;
```

这不仅消除了单独维护程序的复杂性，而且使报表的错误报告机制标准化。

这是菜单模块的 Startup Code(启动代码)属性。在启动加载菜单的表单时，Oracle Developer执行在这个属性中的任何代码。这个代码可以在某些少有的情况下，即在显示表单以前做某些与菜单有关的事情的情况下，完成各种建立任务。在可能的情况下应该依赖缺省的属性，但有时，必须根据平台或某些类似的事情给不同的菜单项赋能。

4. 利用特殊的菜单项

Menu Item Type属性用来改变要显示的菜单项的种类。有五种类型的菜单项，如表 6-7所示。

表6-7 Menu Item Type属性值

值	说 明
Plain	标签缺省类型
Check	显示标签检查记号；设置项目状态为TRUE或FALSE,可通过具有CHECKED选项的GET_MENU_ITEM_PROPERTY内部函数对其进行测试并通过 SET_MENU_ITEM_PROPERTY进行设置
Radio	制作项目为单选按钮组中项目集之一，从此选择一个为 TRUE的项目集并将其余没为FALSE；然后可通过GET_MENU_ITEM_PROPERTY测试状态并通过SET_MENU_ITEM_PROPERTY进行设置
Separator	制作项目分隔符行以建立可视项目组
Magic	几种特殊菜单项目之一：Cut、Copy、Paste、Clear、Undo、About、Help、Quit和Window

可以将一个命令与检查和无线电菜单项相结合，但这样使用的大多数将产生不直观的作用，所以应该避免这样做。在检查一个复选框时，检查菜单项应该呈现相同的特性。接通和断开一个选项，不是启动一个进程而是设置一个属性。大多数用户界面比如 Windows界面的指南强调这些类型菜单项的特性：An Application Design Guide(应用程序设计指南))微软出版社1992)。

魔术般的Cut(剪切)、Copy(复制)、Paste(粘贴)、Clear(清除)、Quit(退出)和Window在Oracle Developer中都有内部功能性，这些功能也用适合于平台的方式 (位置和类型)处理它们的显示。必须为另外一些项建立命令文本或子菜单，比如 Help、About和Undo。

5. 在表单中建立菜单

要把一个菜单粘贴到表单上，首先编译这个菜单 (File | Administration | Compile或Compile File)，在Object Navigator中选择表单，然后在这个表单的 Menu Module(菜单模块)属性中输入菜单模块名。

当使用Call_Form程序启动表单时，可以告诉 Oracle Developer，使用当前的菜单而不是使用粘贴到要启动的表单中的菜单。这就让不同的表单共享应用程序中的单个菜单。

注意 还可以为菜单建立基于角色的安全性。有关细节请参考第9章的内容。

China-pub.com
下载

6. 按键分配

在表单应用程序中可以使用三种按键分配：标准功能键，菜单访问键和加速键。

前面部分提到过标准功能，比如 Execute_Query(执行查询)或Next_Record(下一个记录)。这些功能有缺省的键设置。可以利用 Oracle Terminal实用程序来改变它们，缺省设置是在菜单的PL/SQL命令文本中使用DO_Key，可以确保菜单命令和功能键有相同的作用。

大多数的GUI平台定义了一组菜单访问键。菜单访问键选择菜单项的按钮顺序 (ALT-F-G, CTRL-D，COMMAND-Q等等)，使用户不直接经过菜单。这样菜单访问键的作用就像用菜单项执行命令的键盘捷径。菜单通常根据平台的标准，利用专门的符号显示菜单访问键。例如，在Windows中，菜单访问键呈现为有下划线的大写字母，通过用ALT开头的字符顺序来执行它，并通过菜单条和子菜单上的访问键导航。例如， Ledger Summary报表菜单项是 ALI-Q-R-L (Query菜单，Report子菜单，Ledger Summary项)。

不用专门去做任何事情就可以得到这些菜单访问键。 Oracle Developer利用标签中的第一个大写字母或者第一个字母 (如果设有大写字母的话)来作为菜单访问键。

如果上面的规则会对菜单标签产生两个或多个相同的菜单访问健，那么只要改变标签，告诉Oracle Developer要使用哪个键。可以用大写字母书写要使用的特定字符，如写成“rEport”而不是“Report”。如果不喜欢有趣的大写，那么在字符的前面放一个 &字符，例如“R&eport”。Oracle Developer除去这个&字符并使紧跟它的字母为菜单访问键。如果希望的话，还可以用两个&字符把一个文字“&”放入菜单项中。

Oracle Developer还提供加速键。加速键是一个键序列，可以把它和它的菜单访问键一起分配给菜单项。通常这样做是为了缩短键访问序列或把标准的键序列粘贴到某种标准的功能上。

Oracle Developer提供五种逻辑加速键：Accelerator1到Accelerator5。用Oracle Terminal来建立(第15章讲述有关细节)。在任何给定的菜单模块中，只能指定五个加速键。要指定某个键，在Object Navigator中找出菜单项，并显示它们的属性设置板。把逻辑名，比如 Accelerator3放入Accelerator属性中，然后利用Oracle Terminal，把这个逻辑键和物理键序列联系起来，比如CTRL-X或SHIFT-CTRL-Y。

本章介绍了在Form Builder中可以利用的高级图形设计技术。利用触发器、在第 5章中的其他控制特性、窗口、在本章学习的定制的菜单，可以建立高级的 GUI应用程序。以下两章把你的能力扩展到报表和图表。后面的章节给出一些专门的工具，使设计更安全、更耐用和可再利用。