CHAPTER 8

Using Oracle8i's Export Utility

In This Chapter

Using Oracle8i's Export utility

Exporting databases

Exporting by user

Exporting by table

Estimating the size of export files

Using export options

Oracle's Export utility allows you to extract data from a database and write that data to an operating system file. Together with the Import utility, which you will read about in the next chapter, this provides you with a convenient way to move data between databases. You can export an entire database, or you can choose to limit the export to objects owned by a specific user or to a specific list of tables. This chapter discusses how you can use the Export utility and all of its options to export data to operating system files.

Using Oracle8i's Export Utility

The file to which you extract data when you use Oracle8i's Export utility is referred to as a *dump file*. Export's dump files contain both metadata and data. *Metadata* refers to the data definition language (DDL) statements necessary to re-create the objects that have been exported. If you export your entire database, for example, the dump file will contain `CREATE TABLE` statements, `GRANT` statements, and everything else necessary to re-create your database. Some of the more important uses for the Export utility include the following:

- Copying tables, or entire schemas, from one database to another.
- Reorganizing a table by exporting the data, re-creating the table with different storage parameters, and reloading the data—all in the same database.
- Storing data as a secondary backup, in case the primary backup fails. This works well only for small databases.
- Creating a logical backup that you can use to restore specific tables rather than the entire database.
- Creating a temporary backup of objects that you are going to delete, just in case you later find that you really do need them.

You almost always use the Export utility in conjunction with the Import utility. About the only time that you ever export data without importing it again is when you are using the Export utility as a backup. Even then, you have to be *prepared* to import it again. The Import utility is described in Chapter 9, "Using Oracle8i's Import Utility." To effectively use the Export utility, you need to know how to do several operations, including:

- ✦ Starting the Export utility
- ✦ Passing parameters to it
- ✦ Running it interactively
- ✦ Getting help when you need it
- ✦ Using its prerequisites

Starting the Export utility

The proper command to start the Export utility depends on the release of and the platform on which you are running Oracle. On UNIX systems, and on Windows systems beginning with the Oracle8i release, you use the `exp` command to start the Export utility. On Windows releases prior to Oracle8i, the command contains the release number, so the command you use — `exp80`, `exp73`, or so on — depends on the specific release of Oracle that you are using.

Export is a command-line utility, so you have to run it from the command prompt. On Windows systems, this means that you need to open a Command Prompt window. The following example shows Export being invoked from a Windows NT command prompt to export all the tables, indexes, and other objects owned by the user named AMY:

```
C:\> exp system/manager@bible_db file=amy log=amy owner=amy

Export: Release 8.1.5.0.0 - Production on Fri Aug 6 11:45:53 1999

(c) Copyright 1999 Oracle Corporation.  All rights reserved.


Connected to: Oracle8i Release 8.1.5.0.0 - Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production
Export done in WE8ISO8859P1 character set and WE8ISO8859P1 NCHAR character set
About to export specified users ...
```

There are two basic modes in which to invoke and use Export. You can invoke it either interactively or from the command line. When you run Export interactively, you enter your username and password, respond to some prompts for information about what you want to export, and then let the utility export the data. While this

mode is easy to use, the interactive interface is limited. It doesn't support all the functionality that Export provides. The second way to invoke Export, the one shown in the previous Windows NT platform example, and the one you should focus your efforts on learning, is to pass information to Export using command-line parameters.

Getting help

Export supports a large number of parameters, and it's difficult to remember the ones that you don't use frequently. To help jog your memory, you can run Export with the `HELP=Y` parameter, causing it to display a brief help screen. An example help screen is shown in Listing 8-1.

Listing 8-1: **Export's online help**

```
C:\>exp help=y

Export: Release 8.1.5.0.0 - Production on Fri Aug 6 11:23:00 1999

(c) Copyright 1999 Oracle Corporation.  All rights reserved.



You can let Export prompt you for parameters by entering the EXP
command followed by your username/password:

     Example: EXP SCOTT/TIGER

Or, you can control how Export runs by entering the EXP command followed
by various arguments. To specify parameters, you use keywords:

     Format:  EXP KEYWORD=value or KEYWORD=(value1,value2,...,valueN)
     Example: EXP SCOTT/TIGER GRANTS=Y TABLES=(EMP,DEPT,MGR)
               or TABLES=(T1:P1,T1:P2), if T1 is partitioned table

USERID must be the first parameter on the command line.

Keyword  Description (Default)       Keyword      Description (Default)
------------------------------------------------------------------------
USERID   username/password           FULL         export entire file (N)
BUFFER   size of data buffer         OWNER        list of owner usernames
FILE     output files (EXPDAT.DMP)   TABLES       list of table names
COMPRESS import into one extent (Y)  RECORDLENGTH length of IO record
GRANTS   export grants (Y)           INCTYPE      incremental export type
INDEXES  export indexes (Y)          RECORD       track incr. export (Y)
ROWS     export data rows (Y)        PARFILE      parameter filename
CONSTRAINTS export constraints (Y)   CONSISTENT   cross-table consistency
```

```
LOG      log file of screen output    STATISTICS   analyze objects (ESTIMATE)
DIRECT   direct path (N)              TRIGGERS     export triggers (Y)
FEEDBACK display progress every x rows (0)
FILESIZE maximum size of each dump file
QUERY    select clause used to export a subset of a table

The following keywords only apply to transportable tablespaces
TRANSPORT_TABLESPACE export transportable tablespace metadata (N)
TABLESPACES list of tablespaces to transport

Export terminated successfully without warnings.
```

You may find that `HELP=Y` is one of the parameters you will use most frequently.

Using Export parameters

The general form for invoking the Export utility looks like this:

```
exp [username[/password[@service]]] [param=value
[param=value]...]
```

Replace *username* and *password* with your username and password. If you omit either of these, Export will prompt you for them. If you are exporting data from a remote database, you can include a Net8 service name after the password. The parameters, shown as *param* in the syntax, are those listed on the help screen. Table 8-1 provides a brief description of each. You can place as many parameters as you need to use on the command line.

Table 8-1
Export Parameters

Parameter	*Description*
BUFFER	Specifies the size of the buffer used to fetch rows from the database. This value is in bytes, and it is ignored for direct-path exports. (Direct-path exports will be discussed later in the chapter.) Larger values lead to better performance. The default value for this parameter is operating-system–specific.

Parameter	*Description*
COMPRESS	Specifies whether you want the data for each table to be loaded into one extent, in the event that you import the data back into the database. The default is Y, causing the data to be compressed. Use N if you don't want to do this.
CONSISTENT	Specifies whether you want *all* the data exported to be consistent with respect to a single point in time. The default is N, meaning that each table is exported as a separate transaction. Use Y to have the entire export done as one transaction. You might do this if you expect users to be changing data while you are exporting it.
CONSTRAINTS	Controls whether table constraints are exported. The default is Y. Use N if you want tables to be exported without constraint definitions.
DIRECT	Controls whether a direct-path export is done. (Direct-path exports will be discussed later in the chapter.) The direct method is much faster than the conventional export path. The default is N, which provides a conventional export. Use DIRECT=Y to do a direct-path export.
FEEDBACK	Specifies for Export to indicate progress by displaying periods on the screen. The default is FEEDBACK=0, resulting in no progress display. The FEEDBACK=10 parameter causes a period to be written for every ten rows written to the export file, FEEDBACK=20 causes a period to be written for every 20 rows, and so forth. Be careful with this parameter. If you have a billion-row table, you could be watching a lot of periods pass by.
FILE	Specifies the name of the export file. The default is FILE=expdat.dmp. The default file extension is .dmp.
FILESIZE	Allows you to spread exported data over multiple files. The value used with FILESIZE places an upper limit on the number of bytes written to a file. The default is FILESIZE=0, causing all data to be written to one file. Here are some other possibilities: FILESIZE = 1024 (Generate 1k files) FILESIZE = 1K (Generate 1k files) FILESIZE = 1M (Generate 1 megabyte files) FILESIZE = 1G (Generate 1 gigabyte files)

Continued

Table 8-1 *(continued)*

Parameter	*Description*
	If an export requires multiple files, you will be prompted for more file names. If you don't want to be prompted, you can include a list of file names with the FILE parameter. For example: FILE=(bibdb_1,bibdb_2,bibdb_3) If you don't specify enough names, Export will prompt you for more.
FULL	Specifies that the entire database is to be exported when the FULL=Y parameter is used. The default is FULL=N.
GRANTS	Specifies that grants are to be exported with tables. Grants allow other users to use the tables. The GRANTS=Y parameter is the default. The GRANTS=N parameter causes tables to be exported without their associated grants.
HELP	Controls the display of the help screen shown earlier in this chapter. The parameter is HELP=Y; there is no HELP=N option.
INCTYPE	Specifies the incremental Export options. You can use the following option values:
INCREMENTAL	Exports all database objects that have changed since the last incremental, cumulative, or complete export
CUMULATIVE	Exports all database objects that have changed since the last cumulative or complete export
COMPLETE	Exports all objects Oracle plans to remove support for these options in a future release, in favor of using Recovery Manager. Recovery Manager is a utility that helps you manage the backup and recovery of your database.
INDEXES	Specifies whether indexes should be exported. The default is INDEXES=Y. Use INDEXES=N if you don't want index definitions to be exported.
LOG	Specifies the name of a log file that will accumulate information about the export, including any error messages. The default file extension is .log.
OWNER	Allows you to export data and objects for a specific user or a list of users.

Parameter	*Description*
PARFILE	Allows you to read export parameters from a file. (This option is discussed later in the chapter.)
QUERY	Specifies exporting only a subset of rows from one or more tables. The value for this parameter must be a WHERE clause, and it is applied to the SELECT statement that Export executes against each table.
RECORD	Indicates whether to record an incremental or cumulative export in the export system tables. The default is RECORD=Y. Use RECORD=N if you want to take an incremental or cumulative export without recording that fact.
RECORDLENGTH	Specifies the length, in bytes, of the records in the export file. This option is useful if you are planning to transfer the export file to an operating system that supports smaller record sizes than the one on which you are doing the export. The default is operating-system-specific.
ROWS	Controls whether or not table data is exported. The default is ROWS=Y, which causes data to be exported. Use ROWS=N if you want to export table definitions, but not the data.
STATISTICS	Specifies the type of database optimizer statistics to generate when you import the data that you are now exporting. The valid options are ESTIMATE, COMPUTE, and NONE. If your are importing tables for which statistics existed prior to the export, then the Import utility will automatically regenerate statistics unless you specify STATISTICS=NONE.
TABLES	Allows you to export a specific table or a list of tables.
TABLESPACES	Allows you to export a specific tablespace or a list of tablespaces. The tablespaces must be locally managed, and you must use this parameter in conjunction with TRANSPORT_TABLESPACE=Y.
TRANSPORT_TABLESPACE	Allows you to export the metadata for transportable tablespaces. The default is TRANSPORT_TABLESPACE=N.
TRIGGERS	Controls whether trigger definitions are exported with tables. The default is TRIGGERS=Y, which causes triggers to be exported. Use TRIGGERS=N to export tables, but not their triggers.
USERID	Specifies the username and password of the user invoking the export.

The examples in this chapter show the more common uses of the Export utility. Many of these examples demonstrate the use of the parameters shown in Table 8-1. The PARFILE parameter is significant because it allows you to read parameters from a file. Placing parameters in a file allows you to save complex exports for use again in the future. You'll see how to do this later in this chapter.

Using interactive mode vs. command-line mode

Although you get full access to Export's functionality only when you invoke the utility using command-line parameters, Export does support a limited interactive mode. To invoke Export interactively, simply start it without passing any parameters. The format to use is:

```
exp [username[/password[@service]]]
```

Although you can't pass any parameters, you can pass in the username and password on the command line. It's best not to place the password on the command line, though, lest other people see it. The example shown in Listing 8-2 shows you how to invoke Export with only the Net8 service name and your username on the command line, so that you will be prompted for the password. The benefit of this is that the password is not displayed when you type it. This example goes on to further export the AQUATIC_ANIMAL table owned by the user named SEAPARK.

Listing 8-2: Invoking the Export utility with a Net8 service name

```
E:\Jonathan\Oracle_Bible\ch8> exp seapark@bible_db

Export: Release 8.1.5.0.0 - Production on Fri Aug 6 13:01:05 1999

(c) Copyright 1999 Oracle Corporation.  All rights reserved.

Password:

Connected to: Oracle8i Personal Edition Release 8.1.5.0.0 - Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production
Enter array fetch buffer size: 4096 >

Export file: EXPDAT.DMP > aquatic_animal

(2)U(sers), or (3)T(ables): (2)U > t

Export table data (yes/no): yes >
```

```
Compress extents (yes/no): yes >

Export done in WE8ISO8859P1 character set and WE8ISO8859P1 NCHAR character set

About to export specified tables via Conventional Path ...
Table(T) or Partition(T:P) to be exported: (RETURN to quit) > aquatic_animal

. . exporting table                  AQUATIC_ANIMAL         10 rows exported
Table(T) or Partition(T:P) to be exported: (RETURN to quit) >

Export terminated successfully without warnings.
```

In this example, defaults were taken for pretty much everything except the file name and the table name. As you can see, the AQUATIC_ANIMAL table was exported, and it would have been written to a file named aquatic_animal.dmp. The Enter key (or Return key on some systems) was pressed at the end to exit the utility.

Normal users can use Export interactively like this to export specific tables or to export their entire schema. As a DBA, you will have the additional options of exporting the entire database and exporting other schemas.

Using Export prerequisites

To use the Export utility, a user must have the CREATE SESSION privilege on the target database. That's all you need as long as you are executing objects that you own. To export tables owned by another user or to export the entire database, you must have the EXP_FULL_DATABASE role, and you must have it enabled. Typically, you will have the DBA role, which includes the EXP_FULL_DATABASE role, so you can export pretty much anything that you want to export.

Some schemas are special, and you can't export them even if you are the DBA. The list of restricted schemas includes SYS, ORDSYS, CTXSYS, MDSYS, and ORDPLUGINS. These schemas are special in that they own software and other objects that are very specific to a database, and to a database release. You could damage another database by importing objects into the SYS schema, for example, so Export simply doesn't allow you to export those objects in the first place.

Before using Export against a database, you must run the CATEXP.SQL script once to create views and tables that the Export utility requires. The EXP_FULL_DATABASE role is one of the items that CATEXP.SQL creates. The CATEXP.SQL script is run by CATALOG.SQL, so if you ran CATALOG.SQL when you first created the database, you're all set. If you find that you do need to run either of these scripts, you'll find them in the $ORACLE_HOME/RDBMS/ADMIN directory.

Exporting Databases

You can export the entire database by using the FULL=Y option. This has the effect of exporting all tables, indexes, and other objects for all users. The example shown in Listing 8-3 demonstrates a full export being done:

Listing 8-3: **Exporting an entire database**

```
E:\ch8> exp system/manager file=bible_db log=bible_db full=y

Export: Release 8.1.5.0.0 - Production on Fri Aug 6 14:23:33 1999

(c) Copyright 1999 Oracle Corporation.  All rights reserved.


Connected to: Oracle8i Release 8.1.5.0.0 - Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production
Export done in WE8ISO8859P1 character set and WE8ISO8859P1 NCHAR character set

About to export the entire database ...
. exporting tablespace definitions
. exporting profiles
. exporting user definitions
. exporting roles
. exporting resource costs

...

. exporting job queues
. exporting refresh groups and children
. exporting dimensions
. exporting post-schema procedural objects and actions
. exporting user history table
. exporting default and system auditing options
Export terminated successfully without warnings.

E:\Jonathan\Oracle_Bible\ch8>
```

Sometimes, you may want to export just the definitions from a database and not the data. One reason you might do this is when you want to make a copy of a database — perhaps for use in testing or development — where you want the same structure but not the same data. Use ROWS=N, as shown in the following example, to prevent any data from being exported, leaving only the object definitions in the export file:

```
exp system/manager file=bible_db log=bible_db full=y rows=n
```

One use for the ROWS=N option, and this is one that Oracle doesn't officially support, is to generate the DDL statements necessary to re-create your objects. Export files are binary files, but you can usually load them into a text editor and clean them up a bit. If you do, you'll find that they contain all the CREATE TABLE, CREATE INDEX, and other statements necessary to re-create your schema objects.

Exporting by User

You can export tables for one specific user or a group of users with the OWNER parameter. The following command, for example, exports all objects owned by the user named SEAPARK:

```
exp system/manager file=seapark log=seapark owner=seapark
```

If you want to export several users at once, place a comma-separated list of usernames within parentheses, as shown in the following example, which exports tables for all the users in the sample database used for this book:

```
exp system/manager file=seapark log=seapark
    owner=(seapark, amy, amyc, harold)
```

One issue to be aware of when exporting a user is that although you get all objects owned by the user you are exporting, you don't get any public synonyms that reference those objects. This can be troublesome if you are attempting to copy a schema from a production database into a test database, and the software that you are testing expects synonyms to be in place.

If you run into this situation, you can use SQL*Plus to generate the needed CREATE PUBLIC SYNONYM commands. The following SQL*Plus commands, for example, will create a script file with commands to re-create all public synonyms that point to objects owned by SEAPARK:

```
SET LINESIZE 132
SET PAGESIZE 0
SET TRIMSPOOL ON
SPOOL C:\A\SEAPARK.SYN
SELECT 'create public synonym '
       || synonym_name
       || ' for '
       || table_owner || '.' || table_name
       || ';'
  FROM dba_synonyms
 WHERE table_owner = 'SEAPARK'
   AND owner = 'PUBLIC';
SPOOL OFF
```

Once you've executed these commands and generated the script, you can connect to the target database and run the script there (using the *@filename* command) to re-create the synonyms.

Exporting by Table

You can export a database, you can export a user, and you can also export a table. Use the TABLES parameter to do this. The following command will export just the TANK table:

```
exp seapark/seapark file=tank log=tank tables=tank
```

In this case, because it is the SEAPARK user running the export, it will be SEAPARK's TANK table that is exported. If you are a privileged user (EXP_FULL_DATABASE), you can export a table owned by another user simply by prefacing the table name with the username, using the standard dot notation. In this example, the SYSTEM user is exporting SEAPARK's TANK table:

```
exp system/manager file=tank log=tank tables=seapark.tank
```

Don't try to use the OWNER parameter to specify the owner when you want to export someone else's table. The OWNER and TABLES parameters are not compatible, and they can't be used in the same command. Use the dot notation as shown in the example.

To export multiple tables, enclose a comma-separated list of table names within parentheses. Consider this example:

```
exp system/manager file=tank log=tank
    tables=(seapark.tank, amy.artist)
```

Exporting and importing a single table gives you a way to reorganize how a table is stored. For example, suppose that you want to move a table from one tablespace to another. Perhaps you accidentally created the table in the SYSTEM tablespace, which is not a good place for user tables. One way to move the table is to export it, drop it, re-create it in the correct tablespace, and then import it back again. Chapter 9, "Using Oracle8i's Import Utility," contains information about the Import utility that you need to know to do this.

Estimating the Size of Export Files

If you're exporting a large table or a large database, you need to give some thought to the size of the export file. Based on size, you can choose to export the file to the appropriate disk. One way to estimate size is to query the DBA_SEGMENTS view. The following query returns the total bytes allocated to all tables in the database:

```
SELECT SUM(bytes)
FROM dba_segments
WHERE segment_type = 'TABLE';
```

Since tables may not use all the space that was allocated to them, the results returned by this query can represent only an approximation. Chances are good that it will be on the high side.

If you are exporting by user or by table, you can further qualify the query so that it returns information only related to the tables that you are exporting. The following two queries return the total bytes used by all tables for the SEAPARK user, and for just the AQUATIC_ANIMAL table, respectively:

```
SELECT SUM(bytes)
FROM dba_segments
WHERE segment_type = 'TABLE'
AND owner='SEAPARK';

SELECT SUM(bytes)
FROM dba_segments
WHERE owner='SEAPARK'
AND segment_type='TABLE'
AND segment_name='AQUATIC_ANIMAL';
```

These queries return information only about table data. Large object data, such as BLOB columns, CLOB columns, and so forth, are often stored in segments of type LOBSEGMENT. If you are exporting tables with large object columns, be sure to query that segment type as well.

Using Export Options

You should be aware of at least five significant export options. Each of these options is associated with a parameter that you can specify on the Export utility's command line. The following five options are discussed in this section:

- ✦ Using export paths
- ✦ Compressing extents
- ✦ Exporting a subset of table data
- ✦ Splitting exports across multiple files
- ✦ Using parameter files

Using export paths

Direct-path exports have been around at least since the Oracle 7.3 days. But while they are nothing new, they do provide a significant performance boost over conventional-path exports. The ability to export a subset of a table's data and the

ability to split exports across multiple files are new with Oracle8i. Oracle's Export utility provides the following two paths for exporting table data:

- ✦ Conventional
- ✦ Direct

The conventional-path export represents the way exports have been done from day one. The direct-path export option is a performance enhancement that was added a few years ago.

Conventional-Path Exports

The conventional-path export retrieves data in exactly the same way that a user program would. You use a SQL `SELECT` statement to retrieve the data for each table being exported. As with any other executed query, the data is read from disk and placed into the database buffer cache. From there, the data is moved to a work area where the rows are pulled out, examined to see if they match the query's `WHERE` clause, and then sent on to the Export utility. Conventional-path exports are the default, although you can use the `DIRECT=N` parameter to explicitly ask for one.

Direct-Path Exports

A direct-path export bypasses the evaluation of the individual rows and extracts data much faster than the conventional path. A SQL query is still issued, and blocks from the table are still read into the database buffer cache, but from there they are passed back directly to the Export utility. Figure 8-1 shows the difference between the two methods.

Direct-path exports do have some limitations. Using the `QUERY` parameter is incompatible with direct-path exports because you have to evaluate each row to export a subset of a table. Direct-path exports also cannot be used when the tables that you are exporting contain LOB columns. However, when you can use the direct-path, the performance impact is significant. Add the `DIRECT=Y` parameter to your export commands to ask for direct-path exports.

Compressing extents

If you have a table with data stored in several small extents, you can use the Export and Import utilities together to reorganize the table's data into one large extent. This reduces fragmentation in the tablespace and can have a positive impact on performance. The process of reorganizing a table looks like this:

1. Export the table using the `COMPRESS=Y` parameter
2. Drop the existing table
3. Import the table back

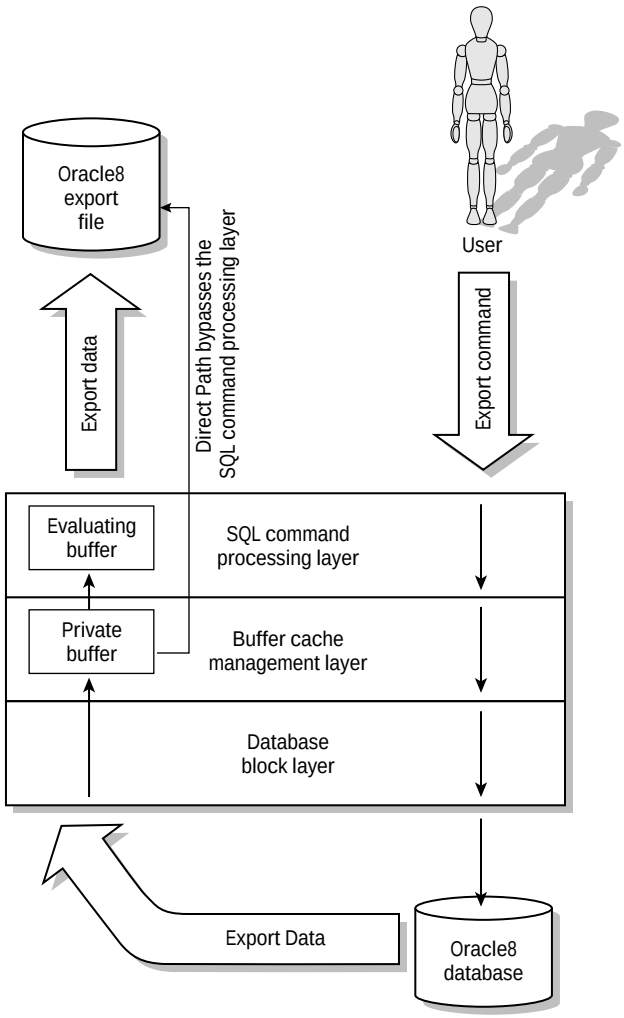


Figure 8-1: Direct-path exports vs. conventional-path exports

The `COMPRESS=Y` parameter, which happens to be the default anyway, causes the Export utility to modify the `INITIAL` storage parameter for the table, and to make it large enough so that when the table is imported back, the initial extent will hold all the data.

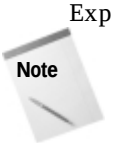
If you have a large table, and the size of that table is greater than the largest block of freespace in your database, then you won't be able to import it as one large extent. In such cases, export the table using `COMPRESS=N`, so that Import won't try to create one large extent when you import the table back.

If you're curious about how extent compression works, pick a table and export it both ways, once with `COMPRESS=Y` and once with `COMPRESS=N`. Be sure to pick a table that is using multiple extents. Use an editor to view both export dump files. The files will contain a lot of binary data, but you should be able to find the `CREATE TABLE` commands in each file. Look at the `STORAGE` clauses. You'll see that the value for `INITIAL` is different.

When many people first learn about `COMPRESS=Y`, it seems strange to them to have to specify this parameter when exporting the data, while the change is really taking place on the import operation. The reason for this is simple. The Export utility generates the DDL to re-create the table, so the Export utility generates the modified `STORAGE` clause. The Import utility simply reads and executes what the Export utility wrote.

Exporting a subset of a table's data

Oracle8i implements an interesting new option that allows you to export only a subset of a table's data. You do this by including a `WHERE` clause as an export parameter. This `WHERE` clause is then appended to the SQL select statement that Export uses to retrieve data from the table.



You cannot use a direct-path export to extract a subset of data.

You use the `QUERY` parameter to pass a `WHERE` clause to the Export utility. The following example shows an export of the `SALAD_TYPE` table that includes only `'FRUIT'`:

```
exp system/manager query='where salad_type=''FRUIT''' 
    tables=amy.salad_type file=fruit log=fruit
```

Using the `QUERY` parameter can get messy because of the way operating systems treat quotes in a command line. This example works for NT. The entire `WHERE` clause is inside a quoted string (single quotes), and the single quotes around `FRUIT` have been doubled so that the operating system will treat them as one quote inside the string and not as the terminating quote. On a UNIX system, you would have to use backslashes to escape the special characters, and you would need to enclose the `WHERE` clause in double quotes. For example:

```
exp system/manager query=\"where salad_type=\'FRUIT'" 
    tables=amy.salad_type file=fruit log=fruit
```

If you are exporting multiple related tables, you can subset all of them in one export operation. The only requirement is that the `WHERE` clause must apply equally to all tables involved in the export. The command in the following example exports the two related tables, `SALAD_TYPE` and `SALAD_TRAY`:

```
exp system/manager query='where salad_type=''FRUIT''' 
    tables=(amy.salad_type,amy.salad_tray) file=fruit log=fruit
```

Since both of these tables contain a SALAD_TYPE column, the WHERE clause applies equally well to each one. By exporting both together like this and using the same WHERE clause, you get a consistent set of related data in the export file. In this case, you have all the fruit salad records.

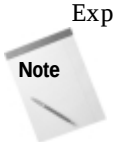
Splitting an export across multiple files

The ability to split an export across multiple files is a new enhancement that became available with the release of Oracle8i. Databases today can be quite large, and as the size of a database increases, so does the size of its export file. In many environments, full exports of production databases are difficult because the resulting files are too large. Many UNIX systems limit file sizes to 2GB or less. If you are exporting a 4GB table, that presents a problem.

Using the new FILESIZE parameter, you can export a large table into several smaller files. For example, if you have a 4GB table named PAYCHECK, and you want to export it into four 1GB files, you can now do that with this command:

```
exp system/manager
    file=(paycheck_1, paycheck_2, paycheck_3, paycheck_4)
    log=paycheck, filesize=1g tables=hr.paycheck
```

The result of this export will be four files named paycheck_1.dmp, paycheck_2.dmp, paycheck_3.dmp, and paycheck_4.dmp. In this example, those four file names were placed in a comma-separated list after the FILE parameter. Export will work through the files one at a time, in the order that you list them. As each file fills to the size you specified, Export will close that file and open the next. If you don't provide enough files in the list for the amount of data being exported, Export will prompt you for more.



If you are running an export like this as a background job, be certain to provide more than enough file names. Otherwise, the export might hang, waiting for you to provide another file name, which you won't be able to do.

Using parameter files

Some of the example export commands in this chapter have been quite long. In real life, the commands can get much larger yet. It's normal to run exports where you list dozens of tables after the TABLES parameter. That's a lot to type on the command line, and if you make a mistake, you have to type it all in again. If you are working with long export lists, or if you want to define export jobs that you can execute repeatedly, you can use the PARFILE parameter to have the Export utility read parameters from a text file.

Suppose that you want to export all the sample tables used for this book, and that you want to be able to do this from time to time without having to rethink how to do it. One solution would be to create a text file with the contents shown in Listing 8-4.

Listing 8-4: **An export parameter file**

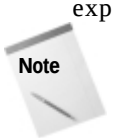
```
# Export the sample tables used for
# the Oracle8i Database Administrator's Bible.

file=bible_tables
log=bible_tables
tables = (
amy.ARTIST
amy.BOOKS
amy.BOOKS_LOANED
amy.BOOKS_RESERVED
amy.BREAD
amy.CLIENT
amy.DAILY_SALES
amy.FISH
amy.MILLIES_MAILING_LIST
amy.MONTHLY_SALES
amy.SALAD_TRAY
amy.SALAD_TYPE
amy.STUDENTS
amy.STUDENTS_FINES
amy.TICKET
seapark.AQUATIC_ANIMAL
seapark.CARETAKER
seapark.CHECKUP
seapark.CHECKUP_HISTORY
seapark.ITEMS
seapark.PARK_REVENUE
seapark.PLAN_TABLE
seapark.TANK
amyc.student
)
```

If the name of this text file was bible_tables.par, you could then issue the following export command whenever you wanted to export the tables:

```
exp system/manager parfile=bible_tables.par
```

The `PARFILE` parameter in this example would tell the Export utility to read parameters from the file bible_tables.par. This is a great way to eliminate a lot of typing and retyping of parameters, as well as to ensure that each subsequent export is consistent with the first.



In a parameter file, pound signs (#) indicate comment lines. Items in a list may either be separated by commas, or they may each be placed on separate lines.

Summary

In this chapter, you learned:

- The Export utility allows you to export a database, a schema, or a table to an operating system file. You can then import the data into other databases, import it back into the same database, or simply hold it as a backup.
- The HELP=Y parameter gets you a concise help screen that lists the other export parameters.
- You must have the EXP_FULL_DATABASE role to export data owned by other users. The DBA role typically includes EXP_FULL_DATABASE.
- Parameter files can be used when you have long parameter lists or when you have export jobs that must be run repeatedly. You use the PARFILE parameter to point the Export utility to a parameter file.
- You can use the QUERY parameter to supply a WHERE clause, which is used to export a subset of data from one or more tables. The WHERE clause that you supply must be applicable to all the tables being exported.
- You use the FILESIZE parameter to limit the size of export files and to spread large exports over multiple files. This makes it possible to export large tables where the export file's size would otherwise be larger than the operating system allows.
- You can reorganize the storage for a table, and you can reduce the number of extents, by exporting the table with COMPRESS=Y, dropping the table, and importing it back again.

✦ ✦ ✦