
CHAPTER# 6

More on SQL

Oracle SQL

- Oracle SQL is a powerful implementation of SQL92
- Oracle SQL*Plus provides a convenient interactive environment with the Oracle database server
- Users can type the commands directly at SQL prompt
- SQL*Plus can execute commands residing in operating system files

Entering and Exiting SQL*Plus

- Enter

sqlplus <userid>/<password>

sqlplus <userid>

sqlplus

- Exit

- The exit command is entered at the SQL prompt

-
- Oracle SQL executed from the SQL*Plus is not a language to build sophisticated applications
 - It is a good language for defining the structure of the database and generating ad-hoc queries
 - To build applications, the power of a full-fledged high level programming language is needed

-
- Once the user is within SQL*Plus environment, the system will usually display the prompt SQL> and wait for user commands:
 - SQL statements to access the database
 - PL/SQL blocks to access the database
 - SQL*Plus commands, for editing and storing SQL statements and PL/SQL blocks, setting options, and formatting query results.

SQLPlus commands

- **Example**

```
CREATE TABLE D
  (D# VARCHAR2(5),
  DNAME VARCHAR2(20),
  STATUS NUMERIC (5),
  LOCATION VARCHAR2(15),
  primary key (D#));
```

-
- Describe: lists the column definitions for a database table

SQL> describe D;

Name	Null?	Type
-----	-----	-----
D#	NOT NULL	VARCHAR2(5)
DNAME		VARCHAR2(20)
STATUS		NUMBER(5)
LOCATION		VARCHAR2(15)

-
- Executes: Executes a single PL/SQL statement
 - Help: online help for SQL*Plus commands
 - Host: executes a host operating system command without leaving SQL*Plus
 - SQL> host ls *.sql
 - ! Can be used instead of host
 - host only without a command will enter the operating system shell. To exit use 'exit'.

-
- remark: used for comments. Also --
 - run: executes SQL statements present in the buffer
 - set: sets SQL*Plus system variables.
 - Set pause on: setting pause to on causes SQL*Plus to pause at the beginning of each page
 - set autocommit on: to inform Oracle to commit any changes to the database immediately after the SQL statement has caused the change is executed

Buffer Manipulation Commands

- The most recent command that is entered on the SQL prompt is stored in the SQL*Plus buffer. It is possible to access, change, append to, and save the contents of this buffer. The SQL*Plus commands for buffer editing are given in the following table.

Command	Abbreviation	Explanation
Append <i>text</i>	<i>A text</i>	Add <i>text</i> to the end of a line
Change <i>/old/new</i>	<i>C /old/new</i>	Change <i>old</i> to <i>new</i> in a line
Change <i>/text</i>	<i>C /text</i>	Delete <i>text</i> from a line
Clear buffer	CI buff	Delete all lines
del		Delete a line
Get <i>file</i>		Load contents of <i>file</i> named file into buffer

Command	Abbreviation	Explanation
input	i	Add one or more lines
Input text	I text	Add a line consisting of text
list	l	List all lines in buffer
List n	L n	List one line and make it the current line
List *	L*	List the current line

Command	Abbreviation	Explanation
List last	L last	List the last line
List m n	L m n	List lines m through n
Save <i>file</i>	Sav <i>file</i>	Save contents of the buffer to file named <i>file</i>

-
- The buffer contents can also be edited using the command “edit”.
 - SQL> edit : it invokes a default operating system editor and loads the buffer contents into the editor
 - SQL> define_editor = vi
 - SQL> define_editor = emacs

Formatting Query results

- SQL*Plus provides commands to format query results that will produce a finished report. The *column* command can be used to change the column heading and to reformat column heading and to reformat column data in a query.

Column <column-name> heading <column-heading>
format <format-model>

column D# heading "Dealer number" format 9999

column DNAME heading "Dealer Name" format A15

column STATUS format 9999

column LOCATION format A15

Screen Capture of an SQL*Plus Session

- Example

```
spool capture.dat
```

```
set echo on
```

```
select D#,LOCATION ,C# from D,DC
```

```
where D.D# = DC.D# and
```

```
city = 'SHELTON';
```

```
set echo off
```

```
spool off
```

```
exit;
```

Drop table

- Usually each create table statement is preceded by a drop table statement. This is a practice that many database programmers and administrators follow to drop any previous version, if there is one, of the table created.
 - drop table <table name>;
 - drop table <table name> cascade constraints;
 - This format is used if there are any foreign keys in other tables referring to the primary or candidate keys of the table being dropped. In this case Oracle drops all referential integrity constraints that refer to primary and unique keys in the dropped table.

Using the Oracle Bulk Loader

- To use the Oracle bulk loader, you need:
 - control file: which specifies *how* data should be loaded into the database
 - data file: which specifies *what* data should be loaded.

Creating the Control File

- A simple control file has the following form:

LOAD DATA

INFILE <dataFile>

APPEND INTO TABLE <tableName>

FIELDS TERMINATED BY '<separator>'

(<list of all attribute names to load>)

-
- **<dataFile>** is the name of the data file. If you did not give a file name extension for **<dataFile>**, Oracle will assume the default extension ".dat". Therefore, it is a good idea to name every data file with an extension, and specify the complete file name with the extension.
 - **<tableName>** is the name of the table to which data will be loaded. Of course, it should have been created already before the bulk load operation.

-
- The optional keyword APPEND says that data will be appended to <tableName>. If APPEND is omitted, the table must be empty before the bulk load operation or else an error will occur.
 - <**separator**> specifies the field separator for your data file. This can be any single character. It is a good idea to use a character that you know will never appear in the data, so the separator will not be confused with data fields.
 - List the names of all attributes of <tableName>, separated by comma and enclosed in parentheses.

- **Example**

```
LOAD DATA  
INFILE test.dat  
INTO TABLE D  
FIELDS TERMINATED BY '|'`  
(D#, DNAME, STATUS, LOCATION)
```

Creating the Data File

- Each line in the data file specifies one tuple to be loaded into <tableName>. It lists, in order, values for the attributes in the list specified in the control file, separated by <separator>. As a concrete example, test.dat might look like:

```
D1|JOHN|40|SHELTON  
D2|MIKEL|20|BRIDGEPORT
```

Loading Your Data

- The Oracle bulk loader is called sqlload. It is a UNIX-level command, *i.e.*, it should be issued directly from your UNIX shell, rather than within sqlplus. A bulk load command has the following form:

```
sqlload userid=<yourName>/<yourPasswd>  
        control=<ctlFile> log=<logFile>
```

-
- `<yourName>` is your Oracle login and `<yourPasswd>` is your Oracle password.
 - The safer form of login, where you give only your login name and let the system prompt you for password applies to `sqlload` as well.
 - `<ctlFile>` is the name of the control file. If no file name extension is provided, `sqlload` will assume the default extension `".ctl"`.

-
- The name of the data file is not needed on the command line because it is specified within the control file.
 - Finally, you may designate <logFile> as the log file. If no file name extension is provided, ".log" will be assumed. sqlload will fill the log file with relevant information about the bulk load operation, such as the number of tuples loaded, and a description of errors that may have occurred.

Loading Without a Separate Data File

- It is possible to use just the control file to load data, instead of using a separate data file.

```
LOAD DATA
```

```
INFILE *
```

```
INTO TABLE D
```

```
FIELDS TERMINATED BY '|'
```

```
(D#, DNAME, STATUS, LOCATION)
```

```
BEGIN DATA
```

```
D1|JOHN|40|SHELTON
```

```
D2|MIKEL|20|BRIDGEPORT
```

Example

- Test.sql file

```
DROP TABLE D;  
CREATE TABLE D  
  ( D# VARCHAR2(5),  
    DNAME VARCHAR2 (20),  
    STATUS NUMERIC (5),  
    LOCATION VARCHAR2(15),  
    primary key (D#));
```

Test.ctl

- Test.ctl

```
LOAD DATA
INFILE test.dat
INTO TABLE D
FIELDS TERMINATED BY '|'
(D#,DNAME,STATUS,LOCATION)
```

- Test.dat file

```
D1|JOHN|40|SHELTON
D2|MIKEL|20|BRIDGEPORT
```

Test.log

- SQL*Loader: Release 8.0.5.0.0 - Production on Sun Apr 23 12:14:0 2000

(c) Copyright 1998 Oracle Corporation. All rights reserved.

Control File: test.ctl

Data File: test.dat

Bad File: test.bad

Discard File: none specified

(Allow all discards)

Number to load: ALL

Number to skip: 0

Errors allowed: 50

Bind array: 64 rows, maximum of 65536 bytes

Continuation: none specified

Path used: Conventional

Table D, loaded from every logical record.

Insert option in effect for this table: INSERT

Column Name	Position	Len	Term	Encl	Data type
S#	FIRST	*			CHARACTER
SNAME	NEXT	*			CHARACTER
STAUS	NEXT	*			CHARACTER
CITY	NEXT	*			CHARACTER

Table SUPPLIER:

2 Rows successfully loaded.

0 Rows not loaded due to data errors.

0 Rows not loaded because all WHEN clauses were failed.

0 Rows not loaded because all fields were null.

Space allocated for bind array: 65016 bytes(63 rows)

Space allocated for memory besides bind array: 0 bytes

Total logical records skipped: 0

Total logical records read: 2

Total logical records rejected: 0

Total logical records discarded: 0

Script started on Sun Apr 23 12:58:00 2000

dbase% sqlplus

SQL*Plus: Release 8.0.5.0.0 - Production on Sun Apr 23 12:58:20 2000

(c) Copyright 1998 Oracle Corporation. All rights reserved.

Enter user-name: elleithy

Enter password:

Connected to:

Oracle8 Enterprise Edition Release 8.0.5.0.0 - Production

PL/SQL Release 8.0.5.0.0 - Production

SQL> select * from D ;

D#	DNAME	STATUS	LOCATION
D1	JOHN	40	SHELTON
2	MIKEL	20	BRIDGEPORT

SQL> exit

Disconnected from Oracle8 Enterprise Edition Release 8.0.5.0.0 - Production

PL/SQL Release 8.0.5.0.0 - Production

dbase% ^D

script done on Sun Apr 23 12:59:36 2000