
3.GETTING STARTED WITH ORACLE8i

- ❑ Creating a table
- ❑ Datatypes
- ❑ Displaying table definition using DESCRIBE
- ❑ Inserting rows into a table
- ❑ Selecting rows from a table
- ❑ Editing SQL buffer
- ❑ Summary
- ❑ Exercises

Creating a Table

A Table is a collection of rows and columns. Data in relational model is stored in tables. Let us create a table first. Then we will understand how to store data into table and retrieve data from the table.

Before a table is created the following factors of a table are to be finalized.

- ❑ What data table is supposed to store.
- ❑ The name of the table. It should depict the content of the table.
- ❑ What are the columns that table should contains
- ❑ The name, data type and maximum length of each column of the table.
- ❑ What are the rules to be implemented to main data integrity of the table.

The following is an example of creation of COURSES table. We actually have six tables in the application that we use throughout the book. We will discuss more about all the tables in the next chapter. But for now, let us create COURSES table and understand how to use basic SQL commands.

The following CREATE TABLE command is used to create COURSES table.

```
SQL> create table COURSES
  2  ( ccode          varchar2(5),
  3    name           varchar2(30),
  4    duration       number(3),
  5    fee            number(5),
  6    prerequisite   varchar2(100)
  7  );
```

Table Created

The above command creates a table called COURSES. This table contains 5 columns. We will discuss about rules to be implemented in this table in the next chapter, where we will recreate this table with all the required rules.

For the time being I want to keep things simple. That is the reason why I am not taking you into constraint and remaining.

Well, we have created our first table. If command is successful, Oracle responds by displaying the message *Table Created*.

Rules to be followed for names

The following are the rules to be followed while naming an Oracle Object. These rules are applicable for name of the table and column.

- ❑ The name must begin with a letter - **A-Z** or **a-z**.
- ❑ Letters, digits and special characters – underscore (**_**), **\$** and **#** are allowed.
- ❑ Maximum length of the name is **30** characters.
- ❑ It must not be an SQL reserved word.
- ❑ There should not be any other object with the same name in your account.

Note: A table can contain up to 1000 columns in Oracle8 or above, whereas in Oracle7 a table can contain only 254 columns.

Datatypes

Each column of the table contains the datatype and maximum length, if it is length is applicable. Datatype of the column specifies what type of data can be stored in the column.

The datatype VARCHAR2 is to store strings that may have different number of characters, NUMBER is used to store numbers. The maximum length, which is given in parentheses after the datatype, specifies how many characters (or digits) the column can store at the most. For example, column VARCHAR2 (20) would mean it can store up to 20 characters.

Table-1 lists out datatypes available in Oracle8i along with what type of data can be stored and maximum length allowed.

Datatype	Description
VARCHAR2(len)	Can store up to len number of characters. Each character would occupy one byte. Maximum width is 4000 characters.
VARCHAR(len)	Same as VARCHAR2. But use VARCHAR2 as Oracle might change the usage of VARCHAR in future releases.
CHAR(len)	Fixed length character data. If len is given then it can store up to len number of characters. Default width is 1. String is padded on the right with spaces until string is of len size. Maximum width is 2000.

NUMBER	Can store numbers up to 40 digits plus decimal point and sign.
NUMBER (p ,s)	P represents the maximum significant digits allowed. S is the number of digits on the right of the decimal point.
DATE	Can store dates in the range 1-1-4712 B.C to 31-12-4712 AD.
LONG	Variable length character values up to 2 gigabytes. Only one LONG column is allowed per table. You cannot use LONG datatype in functions, WHERE clause of SELECT, in indexing and subqueries.
RAW and LONG RAW	Equivalent to VARCHAR2 and LONG respectively, but used for storing byte-oriented or binary data such as digital sound or graphics images.
CLOB, BLOB, NCLOB	Used to store large character and binary objects. Each can accommodate up to 4 gigabytes. We will discuss more about it later in this book.
BFILE	Stores a pointer to an external file. The content of the file resides in the file system of the operation system. Only the name of the file is stored in the column.
ROWID	Stores a unique number that is used by Oracle to uniquely identify each row of the table.
NCHAR (size)	Same as CHAR, but supports national language.
NVARCHAR2 (size)	Same as VARCHAR2, but supports national language.

Table 1: Oracle Datatypes.

Displaying table definition using DESCRIBE

You can display the structure of a table using SQL*PLUS command DESCRIBE. It displays then name, datatype and whether the column can store null value for each column of the table.

The following is the syntax of DESCRIBE command.

```
DESC[RIBE] objectname
```

Displays the column definitions for the specified object. The object may be a table, view, synonym, function or procedure.

To display the structure of COURSES table, enter:

```
SQL> DESC COURSES
Name                               Null?    Type
-----
CCODE                               NOT NULL VARCHAR2 (5)
NAME                                VARCHAR2 (30)
DURATION                             NUMBER (3)
FEE                                   NUMBER (5)
PREREQUISITE                         VARCHAR2 (100)
```

DESCRIBE is an SQL*Plus command and can be abbreviated to DESC.

Inserting rows into a table

Now, let us see how to insert rows into COURSES table. SQL command INSERT is used to insert new row into the table.

While inserting rows, you may enter value for each column of the table or selected columns.

The following command inserts a row into COURSES table.

```
insert into courses
  values('ora','Oracle database',25,4500,'Knowledge of Windows');
```

Note: After inserting the required row, issues COMMIT command to make sure the changes are made permanent. We will discuss more about COMMIT command later in this book but for the time being it is sufficient to know that COMMIT command will make changes permanent. Without COMMIT, rows that are inserted might be lost if there is any power failure.

During insertion, character values are enclosed in single quotes. Unless otherwise specified we have to supply a value for each column of the table. If the value of any column is not known or available then you can give NULL as the value of the column.

For example, the following insert will insert a new row with null value for PREREQUISITE column.

```
insert into courses
  values('c','C Programming',25,3000,null);
```

Note: INSERT command can insert only one row at a time. For multiple row, INSERT command must be issued for multiple times.

DATE type values must be in the format DD-MON-YY or DD-MON-YYYY, where MON is the first three letters of the month (Jan, Feb). If only two digits are given for year then current century is used. For example, if you give 99 for year, Oracle will take it as 2099 as the current century is 2000. So it is important to remember this and give four digits if required.

The following is the complete syntax for INSERT command.

```
INSERT INTO tablename [(columns list)]
  {VALUES (value-1,...) | subquery }
```

We will see how to insert row into a table using a subquery later in this book.

Inserting a row with selected columns

It is possible to insert a new row by giving values only for a few columns instead of giving values for all the available columns.

The following INSERT command will insert a new row only two values.

```
insert into courses(ccode,name)
  values ('odba','Oracle Database Administration');
```

The above command will create a new row in COURSES table with values for only two columns – CCODE and NAME. The remaining columns will take NULL value or the default value, if the column is associated with default value. We will discuss more about default value in the next chapter.

NULL value

Null value means a value that is not available or not known. When a column's value is not known then we store NULL value into the column. NULL value is neither 0 nor blank nor any other known value. We have already seen how to store null value into a column and when Oracle automatically stores null value into a column. We will discuss more about how to process null value later in this chapter.

Selecting rows from a table

Let us see how to retrieve data of a table. SELECT command of SQL is used to retrieve data from one or more tables. It implements operators of relational algebra such as projection, and selection.

The following is the syntax of SELECT command. The syntax given here is incomplete. For complete syntax, please refer to online documentation.

```
SELECT [DISTINCT | ALL]
  { * | table.* | expr } [alias ]
  [ {table}.* | expr } [alias ] ] ...

FROM [schema.]object
  [, [schema.]object ] ...

[WHERE condition]

[ORDER BY {expr|position} [ASC | DESC]
  [, {expr|position} [ASC | DESC]] ...]
```

schema is the name of the user whose table is being accessed. Schema prefix is not required if the table is in the current account. Schema prefix is required while we are accessing a table of some other account and not ours.

The following is an example of a basic SELECT command.

```
select * from courses;
```

CCODE	NAME	DURATION	FEE	PREREQUISITE
ora	Oracle database	25	4500	Windows
vbnet	VB.NET	30	5500	Windows and programming
c	C programming	20	3500	Computer Awareness
asp	ASP.NET	25	5000	Internet and programming
java	Java Language	25	4500	C language
xml	XML Programming	15	4000	HTML,Scripting, ASP/JSP

The simplest SELECT command contains the following:

- ❑ Columns to be displayed. If * is given, all columns are selected.
- ❑ The name of the table from where rows are to be retrieved.

Projection

Projection is the operation where we select only a few columns out of the available columns. The following is an example of projection.

```
select name,fee from courses;
```

NAME	FEE
Oracle database	4500
VB.NET	5500
C programming	3500
ASP.NET	5000
Java Language	4500
XML Programming	4000

Using expressions in SELECT command

It is also possible to include expressions in the list of columns. For example, the following SELECT will display discount to be given for each course.

```
select name,fee, fee * 0.15 from courses;
```

NAME	FEE	FEE*0.15
Oracle database	4500	675
VB.NET	5500	825
C programming	3500	525
ASP.NET	5000	750
Java Language	4500	675
XML Programming	4000	600

Column Alias

The column heading of an expression will be the expression itself. However, as it may not be meaningful to have expression as the result of column heading, we

can give an alias to the column so that alias is displayed as the column heading.

The following example will use alias DISCOUNT for the expression FEE * 0.15.

```
select name, fee, fee * 0.15 DISCOUNT from courses
```

NAME	FEE	DISCOUNT
Oracle database	4500	675
VB.NET	5500	825
C programming	3500	525
ASP.NET	5000	750
Java Language	4500	675
XML Programming	4000	600

The following are the arithmetic operators that can be used in expressions.

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

ORDER BY clause

It is possible to display the rows of a table in the required order using ORDER BY clause. It is used to sort rows on the given column(s) and in the given order at the time of retrieving rows. Remember, sorting takes place on the row that are retrieved and in no way affects the rows in the table. That means the order of the rows will remain unchanged.

Note: ORDER BY must always be the last of all clauses used in the SELECT command.

The following SELECT command displays the rows after sorting rows on course fee.

```
select name, fee from courses order by fee;
```

NAME	FEE
C programming	3500
XML Programming	4000
Oracle database	4500
Java Language	4500
ASP.NET	5000
VB.NET	5500

Note: Null values are placed at the end in ascending order and at the beginning in descending order.

The default order for sorting is ascending. Use option DESC to sort in the descending order. It is also possible to sort on more than one column.

To sort rows of COURSES table in the ascending order of DURATION and descending order of FEE, enter:

```
select name, duration, fee from courses
order by duration , fee desc;
```

NAME	DURATION	FEE
XML Programming	15	4000
C programming	20	3500
ASP.NET	25	5000
Oracle database	25	4500
Java Language	25	4500
VB.NET	30	5500

First, all rows are sorted in the ascending order of DURATION column. Then the rows that have same value in DURATION column will be further sorted in the descending order of FEE column.

Using column position

Instead of giving the name of the column, you can also give the position of the column on which you want to sort rows.

For example, the following SELECT sorts rows based on discount to be given to each course.

```
select name, fee, fee * 0.15
from courses
order by 3;
```

NAME	FEE	FEE*0.15
C programming	3500	525
XML Programming	4000	600
Oracle database	4500	675
Java Language	4500	675
ASP.NET	5000	750
VB.NET	5500	825

Note: Column position refers to position of the column in the selected columns and not the position of the column in the table.

The above command uses column position in ORDER BY clause. Alternatively you can use column alias in ORDER BY clause as follows:

```
select name, fee, fee * 0.15 discount
from courses
order by discount;
```

NAME	FEE	DISCOUNT
C programming	3500	525
XML Programming	4000	600
Oracle database	4500	675
Java Language	4500	675
ASP.NET	5000	750
VB.NET	5500	825

Selection

It is possible to select only the required rows using WHERE clause of SELECT command. It implements *selection* operator of relational algebra.

WHERE clause specifies the condition that rows must satisfy in order to be selected. The following example select rows where FEE is more than or equal to 5000.

```
select name, fee from courses
where fee >= 5000
```

NAME	FEE
VB.NET	5500
ASP.NET	5000

The following relational and logical operators are used to form condition of WHERE clause. Logical operators – AND, OR – are used to combine conditions. NOT operator reverses the result of the condition. If condition returns true, NOT will make the overall condition false.

Operator	Meaning
=	Equal to
!= or <>	Not equal to
>=	Greater than or equal to
<=	Less than or equal to
>	Greater than
<	Less than
AND	Logical ANDing
OR	Logical Oring
NOT	Negates result of condition.

The following SELECT command displays the courses where duration is more than 15 days and course fee is less than 4000.

```
select * from courses
where duration > 15 and fee < 4000;
```

CCODE	NAME	DURATION	FEE	PREREQUISITE
c	C programming	20	3500	Computer Awareness

The following SELECT command retrieves the details of course with code ORA.

```
select * from courses
where ccode = 'ora';
```

CCODE	NAME	DURATION	FEE	PREREQUISITE
ora	Oracle database	25	4500	Windows

Note: When comparing strings, the case of the string must match. Lowercase letters are not equivalent to uppercase letters.

SQL Operators

Apart from standard relational operators (= and >), SQL has some other operators that can be used in conditions.

Operator	What it does?
BETWEEN value-1 AND value-2	Checks whether the value is in the given range. The range is inclusive of the given values.
IN(list)	Checks whether the value is matching with any one of the values given in the list. List contains values separated by comma(,).
LIKE pattern	Checks whether the given string is matching with the given pattern. More on this later.
IS NULL and IS NOT NULL	Checks whether the value is null or not null.

Table 2: SQL Operators.

Now, let us see how to use these special operators of SQL.

BETWEEN ... AND Operator

Checks whether value is in the given range. The range includes all the values in the range including the min and max values. This supports DATE type data also.

To display the list of course where DURATION is in the range 20 to 25 days, enter:

```
select name
from courses
where duration between 20 and 25;
```

NAME

```
Oracle database
C programming
ASP.NET
Java Language
```

Note: BETWEEN.. AND is alternative to using >= and <= operators.

IN Operator

Compares a single value with a list of values. If the value is matching with any of the values given in the list then condition is taken as true.

The following command will retrieve all courses where duration is either 20 or 30 days.

```
select name
from courses
where duration in (20,30);
```

NAME

```
VB.NET
C programming
```

The same condition can be formed even without IN operator using logical operator OR as follows:

```
Select name from courses where duration = 20 or duration = 30;
```

However, it will be more convenient to user IN operator compared with multiple conditions compared with OR operator.

LIKE operator

This operator is used to search for values when the exact value is not known. It selects rows that match the given pattern. The pattern can contain the following special characters.

Symbol	Meaning
--------	---------

%	Zero or more characters can take the place of %.
_ (underscore)	Any single character can take the place of underscore. But there must be one letter.

To select the courses where the course name contains pattern *.NET*, enter:

```
select name,duration, fee from courses
where name like '%.NET%'
```

NAME	DURATION	FEE
VB.NET	30	5500
ASP.NET	25	5000

The following example selects courses where second letter in the course code is "b" and column PREREQUISITE contains word "programming".

```
select * from courses
where ccode like '_b%' and prerequisite like '%programming%';
```

CCODE	NAME	DURATION	FEE	PREREQUISITE
vbnet	VB.NET	30	5500	Windows and programming

Remember LIKE operator is case sensitive. In the above example, if CCODE contains value in uppercase (VB), then it won't be a match to the pattern.

IS NULL and IS NOT NULL operators

These two operators test for null value. If we have to select rows where a column is containing null value or not null value then we have to use these operators.

For example the following SELECT command will select all the courses where the column FEE is null.

```
select * from courses
where fee is null;
```

Though Oracle provides NULL keyword, it cannot be used to check whether the value of a column is null. For example, the following condition will always be false as Oracle treats two null values as two different values.

```
select * from courses
where fee = null;
```

The above command does NOT work as fee though contains null value will not be equal to NULL. SO, we must use IS NULL operator.

Selecting distinct values

DISTINCT clause of SELECT command specifies only distinct values of the specified column must be selected.

The following SELECT command will display only distinct course fee values from COURSES table.

```
select distinct fee from courses;
```

```
      FEE
-----
      3500
      4000
      4500
      5000
      5500
```

Whereas the same query without DISTINCT clause will select the following.

```
select fee from courses;
```

```
      FEE
-----
      4500
      5500
      3500
      5000
      4500
      4000
```

Editing SQL Buffer

Whenever you enter an SQL command in SQL*Plus, it is stored in an area in the memory called as *SQL Buffer*. It is possible to edit the command that is stored in SQL Buffer using a set of commands provided by SQL*Plus. All these commands are SQL*Plus commands.

Note: *SQL*PLUS commands like DESCRIBE are not stored in the buffer. Only SQL commands are stored in the buffer.*

The following list of SQL*PLUS commands are used for editing and other operations related to SQL buffer.

Command	Purpose
A[PPEND] text	Adds text to the end of the current line in the buffer.
DEL	Deletes current line in the buffer.
I[NPUT] [text]	If text is given, then text is placed after the current line, otherwise it allows you to enter a series of lines and places them after the current line.
L[IST]	Displays the contents of buffer.
R[UN]	Runs the current command in the buffer.
/	Same as <i>RUN</i> , but doesn't display command being executed.
SAVE filename	Saves the contents of buffer into <i>filename</i> , which is a file in the host system.
GET filename	Places the contents of <i>filename</i> into buffer.
START filename	Executes commands that are in the given file. The file is also called as <i>start file</i> and <i>command file</i> .
EDIT [filename]	Invokes an editor in the host and places the contents of either given filename or SQL buffer if filename is not given.
HOST command	Executes the command, which is a valid command in the host system.
EXIT	Quits SQL*PLUS.

Table 3: Editing Commands.

Summary

In this chapter we have seen some fundamental SQL commands such as CREATE TABLE, INSERT and SELECT.

SELECT command is the most frequently used command. It is very important to understand how to retrieve required data using SELECT command. We have seen some special operators that are available in SQL. We have seen WHERE and ORDER BY clauses of SELECT command.

Exercises

- _____ is the operator used to compare a column with null value.
- _____ operator is used to compare one value with a set of values.
- The maximum number of characters that can be stored in CHAR type is _____.
- How many LONG columns can a table contain? _____
- In LIKE operator, % stands for _____.
- _____ is used to change the heading of a column.
- SQL commands are to be terminated with _____.
- _____ command is used to display definition of a table.
- Display list of courses where course code starts with letter 'c'.

-
10. Display rows of COURSES table in the ascending order of course fee and descending order of course code.
 11. Select rows from COURSES where course fee is in the range 3000 to 5000.
 12. Add a new row to COURSES table with the following data.
Course code - cpp, name - C++ Programming, duration - 20, fee - 3500, prerequisite - C programming.
 13. Display all the rows where course fee is not known but duration is known.