Department of Information Technology

S.Y. BTech (IT)
SUB: DBMS LAB
Experiment No: 3

Accessing & Modifying Data in Oracle

The Fundamentals of a SELECT Statement

A *SELECT* statement in SQL enables you to retrieve existing data from a Oracle database.

The full syntax of the *SELECT* statement is complex, but the main clauses can be summarized as follows:

- SELECT *select_list*
- [INTO *new_table_name*]
- FROM *table_list*
- [WHERE *search_conditions*]
- [GROUP BY *group_by_list*]
- [HAVING *search_conditions*]
- [ORDER BY *order_list* [ASC | DESC]]

Using Keywords in the Select List

The select list can also contain keywords that control the final format of the result set.

The DISTINCT Keyword

The DISTINCT keyword eliminates duplicate rows from a result set.

```
SELECT DISTINCT ShipCity, ShipRegion
FROM Orders
ORDER BY ShipCity
```

The TOP n Rows

The *FETCH FIRST n Rows only* keyword specifies that the first *n* rows of the result set are to be returned. If ORDER BY is specified, the rows are selected after the result set is ordered. The *n* placeholder is the number of rows to return

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s)
FETCH FIRST number ROWS ONLY;
```

The AS Keyword

You can improve the readability of a *SELECT* statement by giving a table an alias (also known as a correlation name or range variable). A table alias can be assigned either with or without the AS keyword:

■ *table_name* AS *table_alias*
■ *table_name table_alias*

In the following example, the alias p is assigned to the Publishers table:

```
SELECT p.pub_id, p.pub_name
FROM publishers AS p
```

Types of Information in the Select List

A select list can include many types of information, such as a simple expression or a scalar subquery. The following example shows many of the items that you can include in a select list:

```
SELECT FirstName + ' ' + LastName AS "Employee Name",
HomePhone,
Region
FROM Employees
```

The INTO Clause

The INTO clause enables you to specify that the result set will be used to create a new table with the name defined in the clause. A *SELECT…INTO* statement can be used to combine data from several tables or views into one table. You can also use it to create a new table containing data selected from a linked server.

```
SELECT FirstName, LastName
INTO EmployeeNames
FROM Employees
```

The result set that is generated by the statement creates the EmployeeNames table. The new table will contain the FirstName column and the LastName column, and those columns will contain the values from the Employees table.

The WHERE, GROUP BY, and HAVING Clauses

The WHERE and HAVING clauses in a *SELECT* statement control the rows from the source tables that are used to build the result set. The WHERE and HAVING clauses are filters. They specify a series of search conditions, and only those rows that meet the terms of the search conditions are used to build the result set. Those rows that meet the search conditions are said to be qualified to participate in the result set.

```
SELECT CustomerID, CompanyName
FROM Customers
WHERE Region = 'WA'
```

The HAVING clause is typically used in conjunction with the GROUP BY clause, although it can be specified without GROUP BY. The HAVING clause specifies more filters that are applied after the WHERE clause performs its filtering.

The GROUP BY Clause

The GROUP BY keywords are followed by a list of columns, known as the grouping columns. The GROUP BY clause restricts the rows of the result set. There is only one row for each distinct value in the grouping column or columns. Each result set row contains summary data related to the specific value of its grouping columns.
Typically, the HAVING clause is used with the GROUP BY clause, although HAVING can be specified separately.

Processing the WHERE, GROUP BY, and HAVING Clauses

Understanding the correct sequence in which the WHERE, GROUP BY, and HAVING clauses are applied helps in coding efficient queries:

■ The WHERE clause is used to filter the rows that result from the operations specified in the FROM clause.
■ The GROUP BY clause is used to group the output of the WHERE clause.
■ The HAVING clause is used to filter rows from the grouped result.

The ORDER BY Clause

The ORDER BY clause sorts a query result by one or more columns (up to 8060 bytes). A sort can be *ascending* (ASC) or *descending* (DESC). If neither is specified, ASC is assumed. If more than one column is named in the ORDER BY clause, sorts are nested.
The following statement sorts the rows in the Titles table, first by publisher (in descending order), then by type (in ascending order within each publisher), and finally by price (also ascending, because DESC is not specified):

```
SELECT Pub_id, Type, Title_id, Price
FROM Titles
ORDER BY Pub_id DESC, Type, Price
```

To retrieve all data from the Titles table

```
SELECT * FROM Titles
```

To retrieve data from specific columns in the Titles table

```
SELECT Title_id, Title, Price, Ytd_sales
FROM Titles
```

To specify the condition that the result set must meet

```
SELECT Title_id, Title, Price, Ytd_sales
FROM Titles
WHERE Price > 10
```

To specify the order in which the result set appears

```
SELECT Title_id, Title, Price, Ytd_sales
FROM Titles
WHERE Price > 10
ORDER BY Price DESC, Title
```

To group data in a result set

```
SELECT Type, AVG(Price) AS AvgPrice
FROM Titles
WHERE Price > 10
GROUP BY Type
ORDER BY AvgPrice DESC
```

To create a table for the result set

```
SELECT Type, AVG(Price) AS AvgPrice
INTO TypeAvgPrice
FROM Titles
WHERE Price > 10
GROUP BY Type
ORDER BY AvgPrice DESC
```

Using a SELECT Subquery to Add Data

You can use a SELECT subquery in the *INSERT* statement to add values to a table from one or more other tables or views. A subquery enables you to add more than one row at a time.

A SELECT subquery in an *INSERT* statement is used to add subsets of existing data to a table, whereas a VALUES clause is used in an *INSERT* statement to add new data to a table.

```
INSERT INTO NewBooks (BookTitle, BookType)
SELECT Title, Type
FROM Titles
WHERE Type = 'mod_cook'
```

This *INSERT* statement uses the output of the SELECT subquery to provide the data that will be inserted into the NewBooks table.

Modifying the structure of table:

Adding new column:

Syntax:

ALTER TABLE < table name >
ADD (<new column name><data type> (<size>) ;
<New column name><data type> (<size>…);

Example:

Add the field Telephone No a number data type field that can hold a number up to 8 Digits in length.

ALTER TABLE BANK,
ADD(Telephone number (8));

Dropping a column from a table:
Syntax:
ALTER TABLE <table name>
DROP COLUMN <column name>;
Example:

Alter table Bank DROP column Telephone No;

Note that using ALTER TABLE you cannot change:

1. the name of the table
2. the name of the column
3. decrease the size of a column if table data exists.

Implementation in SQL:
Referring to the tables developed in Experiment no 2 solve the following queries.

a. Find out the customers who stay in an area 'SA', or area 'BI' or area 'CH'.

b. List the movies in sorted order of their titles.

c. Calculate the total price of all the movies.

d. Determine the maximum and minimum movie prices. Rename the columns headings as MAXIMUM and MINIMUM while displaying the output..

e. Find the number of movies of each type.

f. Print the type and average price of each type.

g. Calculate the average price for each type that has average price > 150.

h. Calculate the average price of all movies where type is 'comedy' or 'thriller and price>=150

i. Print the names of all customers whose customer id is between A01 and A05

j. List the various movie types available from the movie table.

k. Find the movies whose price is greater than 150 and less than or equal to 200.

l. Retrieve the top 5 customers.

m. Retrieve the top 5 customers in the alphabetical order of first name.

n. Alter the customer table to add the age of every customer.

o. Create a table 'NewCustomer'.Insert the last names and first names of all the customers into this table using select subquery.

p. Print the information of invoice table in the following format for all records :

The Invoice No. Of Customer Id. {cust_id} is {inv_no} and Movie No. Is {movie_no}.