Database Management System (CS-2004) Lab

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

School of Computer Engineering

4 Credit

Mr. Rajat Kumar Behera - Associate Professor

Lab Contents

Sr #	Major and Detailed Coverage Area	Lab#
1	DQL – Part 1 ❑ Recap on DML ❑ SELECT, FROM, WHERE & ORDER BY Clause ❑ Counting records in a Table ❑ Column Aliases ❑ Operators ❑ Summarization	3

Recap on DML

- The **DELETE** statement is used to delete rows from a table. **Example** -
  - **DELETE** FROM EMPLOYEE WHERE ID = 100;
  - **DELETE** FROM EMPLOYEE;
- The **TRUNCATE** statement is used to delete all the rows from the table and **free the space containing the table**. It is **DDL** in the sense that it commits and don't roll back. When we are using truncate, we are de-allocating the whole space allocated by the data without saving into the undo-table-space. But, in case of delete, we are putting all the data into undo table-space and then we are deleting all the data. Example – **TRUNCATE** TABLE EMPLOYEE;

ROWID

For each row in the database, the ROWID pseudo column (Oracle assigned field and not part of the table) returns the address of the row. Oracle rowid values contain information necessary to locate a row. It contains the physical address of a row in a database. If the user wants to delete the duplicate copies of the same record, then ROWID is used. **Example** - SELECT ROWID, Cid FROM CUSTOMER;

ROWID	Cid
AAAF4YAABAAAHCKAAA	1
AAAF4YAABAAAHCKAAB	2
AAAF4YAABAAAHCKAAC	2
AAAF4YAABAAAHCKAAD	3

Recap on DML

- User can view all the constraints by executing **SELECT * FROM USER_CONSTRAINTS;**
- If the user wants to view all the constraints applied to a single table, the **syntax** is: **SELECT * FROM USER_CONSTRAINTS WHERE TABLE_NAME= tablename;**

  Example - **SELECT * FROM USER_CONSTRAINTS WHERE TABLE_NAME='ITEM_MASTER';**

DUAL Table

DUAL table is a small worktable, which consists of only one column **DUMMY** and a single row with value **X** of VARCHAR2 type. This table is owned by user **SYS** and is available to all users. It is used for arithmetic calculations and date retrieval. Example –

- SELECT 2*5 FROM DUAL;
- SELECT SYSDATE FROM DUAL;

DQL

A **query** is an inquiry into the database using the **SELECT** statement. A query is used to extract data from the database in a readable format according to the user's request. For instance, if you have an employee table, you might issue an SQL statement that returns the employee who is paid the most. This request to the database for usable employee information is a typical query that can be performed in a relational database.

The SELECT statement, the command that represents **Data Query Language (DQL)**, is the basic statement used to construct database queries. The **SELECT** statement is not a standalone statement, which means that one or more additional clauses (elements) are required for a syntactically correct query. The **FROM** clause is a mandatory clause and must always be used in conjunction with the SELECT statement. There are four keywords, or clauses, that are valuable parts of a SELECT statement. These keywords are as follows:

- SELECT
- FROM
- WHERE
- ORDER BY

Reference Table…

- **EMPLOYEE** table with the attributes:

  ID, LAST_NAME, FIRST_NAME, MIDDLE_NAME, FATHER_NAME, MOTHER_NAME, SEX, HIRE_DATE, ADDRESS, CITY, STATE, ZIP, PHONE, PAGER
- **SCHOOL** table with the attributes:

  ID, NAME
- **EMPLOYEE _ALIGNMENT** table with the attributes:

  EMPLOYEE_ID, SCHOOL_ID
- **JOB** table with the attributes:

  ID, NAME, TITLE, SALARY, BONUS
- **EMPLOYEE _PAY** table with the attributes:

  EMPLOYEE_ID, JOB_ID
- **PRODUCT** table with the attributes:

  ID, NAME, DESC, STATUS, CREATED_DATE, CREATED_BY, LAST_MODIFIED_DATE, LAST_MODIFIED_BY, COST, INJECTED_DATE

SELECT Clause

The **SELECT** statement is used in conjunction with the **FROM** clause to extract data from the database in an organized, readable format. The **SELECT** part of the query is for selecting the data you want to see according to the columns in which they are stored in a table.

The syntax for a simple SELECT statement is as follows:

** means all column*

TABLE 2 is optional

*SELECT [* | DISTINCT COLUMN1, COLUMN2] FROM TABLE1 [, TABLE2];*

Example: SELECT * FROM PRODUCT;

The DISTINCT option can be used to suppress the display of duplicate records.

Example: SELECT DISTINCT COST FROM PRODUCT;

FROM & WHERE Clause

The **FROM** clause must be used in conjunction with the SELECT statement. It is a required element for any query. The FROM clause's purpose is to tell the database what table(s) to access to retrieve the desired data for the query. The FROM clause may contain one or more tables. The FROM clause must always list at least one table.

WHERE clause is a condition is part of a query that displays selective information as specified by the user. The value of a condition is either TRUE or FALSE, thereby limiting the data received from the query. The WHERE clause places conditions on a query by eliminating rows that would normally be returned by a query without conditions. Syntax is:

[represents optional

*select [* | distinct column1, column2] from table1 [, table2] where [condition1 | expression1] [and | OR condition2 | expression2]*

Example:

SELECT * FROM PRODUCT;

SELECT * FROM PRODUCT WHERE COST < 5;

SELECT * FROM PRODUCT WHERE COST < 5 **AND** ID > 222;

ORDER BY Clause

Data can be **sorted** by using the **ORDER BY** clause. The **ORDER BY** clause arranges the results of a query in a **specified sorted format**. The default ordering of the ORDER BY clause is an **ascending order**; the sort displays in the order A–Z if it's sorting output names alphabetically. A descending order for alphabetical output would be displayed in the order Z–A. Ascending order for output for numeric values between 1 and 9 would be displayed 1–9; descending order would be displayed as 9–1. Syntax is:

[represents optional

*SELECT [* | DISTINCT COLUMN1, COLUMN2] FROM TABLE1 [, TABLE2] WHERE [CONDITION1 | EXPRESSION1] [AND | OR CONDITION2 | EXPRESSION2] ORDER BY COLUMN1 | INTEGER [ASC | DESC]*

Example:

SELECT ID, DESC, COST FROM PRODUCT WHERE COST < 20 **ORDER BY ID ASC;**

SELECT ID, DESC, COST FROM PRODUCT WHERE COST < 20 **ORDER BY 3 DESC;**

Class Work

- Write a SELECT statement that returns the name and cost of each product.
- Write a SELECT statement that returns the name and cost of each product in ascending order.
- Write a query that generates a list of all employee name and their phone numbers.
- Write a query that generates a list of all employee in descending order of their city.

Counting the Records in a Table

A simple query can be issued on a table to get a quick count of the number of records in the table or the number of values for a column in the table. A count is accomplished by the function **COUNT.**

The syntax of the COUNT function is :

SELECT COUNT(*) FROM TABLE_NAME;

Example: SELECT COUNT(*) FROM PRODUCT;

If you want to **count only the unique values** that show up within a table, you would use the **DISTINCT** syntax within the COUNT function. For example, if you want to get the distinct states represented in the STATE column of the EMPLOYEE, use a query such as the one that follows:

SELECT COUNT(DISTINCT STATE) FROM EMPLOYEE;

Column Aliases

Column aliases are used to **temporarily rename a table's columns** for the purpose of a particular query. The following syntax illustrates the use of column aliases:

*SELECT COLUMN_NAME **AS** ALIAS_NAME FROM TABLE_NAME;*

Example:

SELECT ID, DESC **AS** PROD_DESC FROM PRODUCT;

SELECT ID, DESC **AS** "Prod Description" FROM PRODUCT;

Class Work

- Select distinct last name of the employee with appropriate aliases
- Select distinct first name of the employee with appropriate aliases

Operators

An operator is a reserved word or a character used primarily in an SQL statement's **WHERE** clause to perform operation(s), such as comparisons and arithmetic operations. Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.

Commonly used operators are:

- Comparison operators
- Logical operators
- Operators used to negate conditions
- Arithmetic operators

Comparison Operators

Comparison operators test single values in an SQL statement. The comparison operators discussed consist of =, <>, <, and >. These operators are used to test:

- Equality
- Non-equality
- Less-than values
- Greater-than values

Equality

The equal operator compares single values to one another in an SQL statement.

The equal sign (=) symbolizes equality. When testing for equality, the compared values must match exactly, or no data is returned. If two values are equal during a comparison for equality, the returned value for the comparison is TRUE; the returned value is FALSE if equality is not found. This Boolean value (TRUE/FALSE) is used to determine whether data is returned according to the condition. This operator can be used by itself or combine with other operators.

Example - SELECT * FROM PRODUCT WHERE ID = '2345';

Comparison Operators cont…

Non-equality

Opposite of equality operator

Example - SELECT * FROM PRODUCT WHERE ID <> '2345';

SELECT * FROM JOB WHERE SALARY <> 20000

Less Than, Greater Than

Example –

SELECT * FROM PRODUCT WHERE COST > 20;

SELECT * FROM PRODUCT WHERE COST < 20;

Class Work

- Select the employees who are not staying in "Bhubaneswar"
- Select the employees who are staying in "Lucknow"

Logical Operators

Logical operators are those operators to make comparisons. Commonly used operators are:

- IS NULL
- IN
- EXISTS
- BETWEEN
- LIKE

IS NULL

The NULL operator is used to compare a value with a NULL value.

Example –

SELECT ID, LAST_NAME, FIRST_NAME, PAGER FROM EMPLOYEE WHERE PAGER **IS NULL**;

BETWEEN

The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value. The minimum and maximum values are included as part of the conditional set. **Example –**

SELECT * FROM PRODUCT WHERE COST BETWEEN 5.95 AND 14.5;

Logical Operators cont…

IN

The **IN** operator compares a value to a list of literal values that have been specified. For TRUE to be returned, the compared value must match at least one of the values in the list. **Example –**

SELECT * FROM PRODUCT WHERE ID IN (‘13’, ’9’, ’87’, ’119’);

LIKE

The **LIKE** operator is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the LIKE operator:

- The percent sign (**%**) - represents **zero, one, or multiple** characters.
- The underscore (_) - represents a **single** number or character

Example –

SELECT DESC FROM PRODUCT WHERE DESC LIKE ‘**%**S’;

SELECT DESC FROM PRODUCT WHERE DESC LIKE ‘**_**S’;

Logical Operators cont...

EXISTS

The **EXISTS** operator is used to search for the presence of a row in a specified table that meets certain criteria.

Example –

SELECT COST FROM PRODUCT WHERE **EXISTS** (SELECT COST FROM PRODUCT WHERE COST > 100);

Conjunctive Operators

Conjunctive operators provide a means to make multiple comparisons with different operators in the same SQL statement. These operators are:

- AND
- OR

AND

The **AND** operator allows the existence of multiple conditions in an SQL statement's WHERE clause. For an action to be taken by the SQL statement, whether it be a transaction or query, all conditions separated by the AND must be TRUE.

Example –

- SELECT * FROM PRODUCT WHERE COST > 10 AND COST < 30;
- SELECT * FROM PRODUCT WHERE ID = '7725' AND ID = '2345';

Conjunctive OR Operator

The **OR** operator combines multiple conditions in an SQL statement's WHERE clause. For an action to be taken by the SQL statement, whether it is a transaction or query, at least one of the conditions that are separated by OR must be TRUE. **Example –**

- SELECT * FROM PRODUCT WHERE ID = '90' OR ID = '2345';
- SELECT * FROM PRODUCT WHERE COST > 10 AND (ID = '222' OR ID = '90' OR ID = '11235');

Negative Operators

The NOT operator reverses the meaning of the logical operator with which it is used. The NOT can be used with other operators to form the following methods:

- <>, != (NOT EQUAL)
- NOT BETWEEN
- NOT IN
- NOT LIKE
- IS NOT NULL
- NOT EXISTS

Arithmetic Operators

Arithmetic operators perform mathematical functions in SQL—the same as in most other languages. The four conventional operators for mathematical functions are :

- + (addition)
- - (subtraction)
- * (multiplication)
- / (division)

Addition

Addition is performed through the use of the plus (+) symbol.

Example - SELECT SALARY + BONUS FROM JOB;

SELECT SALARY FROM JOB WHERE SALARY + BONUS > 40000;

Subtraction

Subtraction is performed through the use of the minus (-) symbol.

E.g. SELECT SALARY - BONUS FROM JOB;

SELECT SALARY FROM JOB WHERE SALARY - BONUS > 40000;

Arithmetic Operators cont…

Multiplication

Addition is performed through the use of the asterisk (*) symbol.

Example -

SELECT SALARY * 10 FROM JOB;

SELECT ID, SALARY * 1.1 FROM JOB WHERE BONUS IS NOT NULL;

Division

Division is performed through the use of the slash (/) symbol.

Example -

SELECT SALARY / 10 FROM JOB;

SELECT SALARY FROM JOB WHERE (SALARY / 10) > 40000;

Class Work – PRODUCT Table

- Write a SELECT statement that returns product IDs and description (alpha order) for the product, created by Indiana D or Ohio D or Michigan R, or Illinois M.
- Write a SELECT statement that returns product IDs and description (alpha order) for the product, created by Indiana D or Ohio D **and** updated by Michigan R or Illinois M.
- Write a SELECT statement that returns the ID, description, and cost. Limit the product cost to between 1.00 and 12.50 using conjunctive and comparison operator
- Write a SELECT statement that returns the ID, description, and cost. Limit the product cost to between 15.00 and 100.50 using between operator
- SELECT statement that returns description, product cost, and 5% sales tax for each product. List the products in order from most to least expensive.
- Write a SELECT statement that returns description, cost, 5% sales tax for each product and total cost with sales tax. List the products in order from most to least expensive.
- Write a query to return all the products that start with the letter P.
- Write a query to return all products that do not start with the letter P.

Data Summarization

Functions are keywords in SQL used to manipulate values within columns for output purposes. A function is a command normally used in conjunction with a column name or expression that processes the incoming data to produce a result. SQL contains several types of functions. An aggregate function provides summarization information for an SQL statement, such as counts, totals, and averages. The basic set of aggregate functions are:

- COUNT
- MAX
- AVG
- SUM
- MIN

COUNT

COUNT function is to count rows or values of a column that do not contain a **NULL** value. When used within a query, the COUNT function **returns a numeric value**. One can also use the COUNT function with the DISTINCT command to only count the distinct rows of a dataset. **Example** –

```
SELECT COUNT(DISTINCT SALARY) FROM JOB;
SELECT COUNT(ALL SALARY)FROM JOB;
SELECT COUNT(*) FROM EMPLOYEE;
```

Data Summarization cont…

SUM

The SUM function returns a total on the values of a column for a group of rows. You can also use the SUM function in conjunction with DISTINCT. When you use SUM with DISTINCT, only the distinct rows are totaled, which might not have much purpose. Your total is not accurate in that case because rows of data are omitted. **Example –**

SELECT **SUM**(SALARY) FROM JOB;

SELECT **SUM(DISTINCT** SALARY) FROM JOB;

AVG

The AVG function finds the average value for a given group of rows. When used with the DISTINCT command, the AVG function returns the average of the distinct rows. **Example –**

SELECT **AVG**(SALARY) FROM JOB;

SELECT **AVG(DISTINCT** SALARY) JOB;

Data Summarization cont…

MAX

The MAX function returns the maximum value from the values of a column in a group of rows. NULL values are ignored when using the MAX function. The DISTINCT command is an option. However, because the maximum value for all the rows is the same as the distinct maximum value, DISTINCT is useless. **Example –**

SELECT **MAX**(SALARY) FROM JOB;

SELECT **MAX(DISTINCT** SALARY) FROM JOB;

SELECT **MAX**(COST) FROM PRODUCT;

MIN

The MIN function returns the minimum value of a column for a group of rows. NULL values are ignored when using the MIN function. The DISTINCT command is an option. However, because the minimum value for all rows is the same as the minimum value for distinct rows, DISTINCT is useless. **Example –**

SELECT **MIN**(COST) FROM PRODUCT;

Class Work

Consider the table JOB and construct the following queries

- What is the average salary?
- What is the maximum bonus?
- What are the total salaries?
- What is the minimum salary?
- How many rows are in the table?
- Determine how many employees whose last names begin with a G.

Thank You
End of Lab 3

Assignment

1. Create following tables and insert appropriate rows

- **EMPLOYEE** table with the attributes:

  ID, LAST_NAME, FIRST_NAME, MIDDLE_NAME, FATHER_NAME, MOTHER_NAME, SEX, HIRE_DATE, ADDRESS, CITY, STATE, ZIP, PHONE, PAGER

- **SCHOOL** table with the attributes:

  ID, NAME

- **EMPLOYEE _ALIGNMENT** table with the attributes:

  EMPLOYEE_ID, SCHOOL_ID

- **JOB** table with the attributes:

  ID, NAME, TITLE, SALARY, BONUS

- **EMPLOYEE _PAY** table with the attributes:

  EMPLOYEE_ID, JOB_ID

- **PRODUCT** table with the attributes:

  ID, NAME, DESC, STATUS, CREATED_DATE, CREATED_BY, LAST_MODIFIED_DATE, LAST_MODIFIED_BY, COST, INJECTED_DATE

Assignment

2. Retrieve the row count in “Associate Professor” title
3. Retrieve the row count in “Associate Professor” title whose salary is between 80,000 and 100,000
4. Retrieve the details of jobs whose sum of salary and bonus is falling in the range 80,000 and 100,000
5. What is the average salary?
6. What is the maximum bonus?
7. What are the total salaries?
8. What is the minimum salary?
9. How many rows are in the table?
10. Determine how many employees whose last names begin with a G.
11. Select the employees who are not staying in “Bhubaneswar”
12. Select the employees who are staying in “Lucknow”
13. Write a SELECT statement that returns product IDs and description (alpha order) for the product, created by Indiana D or Ohio D or Michigan R, or Illinois M.

Assignment

14. Write a SELECT statement that returns product IDs and description (alpha order) for the product, created by Indiana D or Ohio D and updated by Michigan R or Illinois M.
15. Write a SELECT statement that returns the ID, description, and cost. Limit the product cost to between 1.00 and 12.50 using conjunctive and comparison operator
16. Write a SELECT statement that returns the ID, description, and cost. Limit the product cost to between 15.00 and 100.50 using between operator
17. SELECT statement that returns description, product cost, and 5% sales tax for each product. List the products in order from most to least expensive.
18. Write a SELECT statement that returns description, cost, 5% sales tax for each product and total cost with sales tax. List the products in order from most to least expensive.
19. Write a query to return all the products that start with the letter P.
20. Write a query to return all products that do not start with the letter P.