NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES

CL 203-Database Systems Lab

Lab Session 01

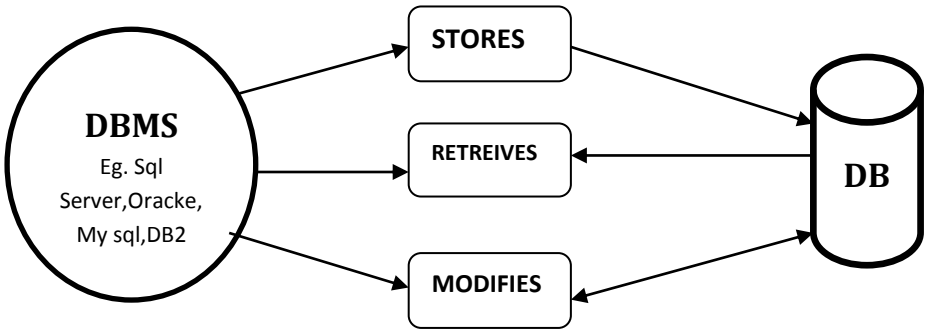
DATABASE

"Collection of inter-related data in an organized manner"

DBMS

"A program that manages Database"

Role of DBMS



What is a Relational Database?

"Collection of relations (tables) or 2-dimensional Tables"

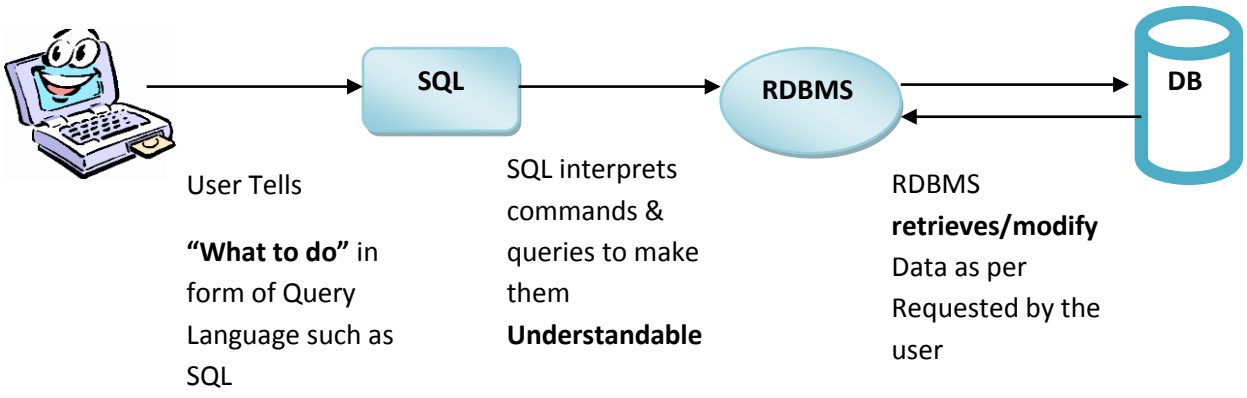
Points to Remember:

- A table is a basic storage structure unit of an RDBMS
- Easy to use
- Flexible in structure
- Security and Authorization methods are well defined
- Protect Data integrity
- Can be accessed and modifies by executing structured query language statements
- Uses a set of relational Operators**(Selection,Projection,Join)** and a set operation **Union, Intersection** etc
- Contains a collection of tables with **No Physical Pointers** as we use **Primary Key & Foreign Key** to access and relate data
- Keeps logical representation of data independent of its physical storage characteristics

Transact Structured Query Language

(T-SQL)

- In most RDBMS, SQL is used as a **language interpreter**



- It is a non-procedural Language i.e. User Only tell **"What To Do"** not **"How To Do"**
- SQl is used for:
  - Data Manipulation
  - Data Definition
  - Data Administration
  - All are expressed as an SQL statement or command.

IMPORTANT

A query is a user request to retrieve data or information with a certain condition.

Using SQL * Plus:

Sql * plus enables you to conduct a **"conversation"** with the database because you can enter SQL statements and View results returned by the database.

Version of SQL* Plus

Windows Version

Programs→Oracle App Dev→SQL * Plus

User Name:----------

Password:----------

Host String:----------

Command-Line Version

Sql Plus[User_name[/password[@host-string]]]

How To...	SQL*Plus Command

Log in to SQL*Plus	`SQLPLUS [ ( username[/password][@connect_identifier] | / )`
	`[ AS ( SYSDBA | SYSOPER ) ]`
	`| /NOLOG`
	`]`

List help topics available in SQL*Plus	`HELP [ INDEX | topic ]`
Execute host commands	`HOST [ command ]`
Show SQL*Plus system variables or environment settings	`SHOW ( ALL | ERRORS | USER | system_variable | ... )`
Alter SQL*Plus system variables or environment settings	`SET system_variable value`
Start up a database	`STARTUP PFILE = filename` `[ MOUNT [ dbname ] | NOMOUNT | ... ]`

Connect to a database	`CONNECT [ [ username [ /password ] [ @connect_identifier ]`
	`[ / AS ( SYSOPER | SYSDBA ) ]`
	`]`

List column definitions for a table, view, or synonym, or specifications for a function or procedure	`DESCRIBE [ schema. ] object`
Edit contents of the SQL buffer or a file	`EDIT [ filename [ .ext ] ]`
Get a file and load its contents into the SQL buffer	`GET filename [ .ext ] [ LIST | NOLLIST ]`
Save contents of the SQL buffer to a file	`SAVE filename [ .ext ] [ CREATE | REPLACE | APPEND ]`

List contents of the SQL buffer	`LIST [ n | n m | n LAST | ... ]`
Delete contents of the SQL buffer	`DEL [ n | n m | n LAST | ... ]`
Add new lines following current line in the SQL buffer	`INPUT [ text ]`
Append text to end of current line in the SQL buffer	`APPEND text`
Find and replace first occurrence of a text string in current line of the SQL buffer	`CHANGE sepchar old [ sepchar [ new [ sepchar ] ] ]` `sepchar can be any non-alphanumeric character such as "/" or "!"`
Capture query results in a file and, optionally, send contents of file to default printer	`SPOOL [ filename [ .ext ]` `[ CREATE | REPLACE | APPEND | OFF | OUT ]`

Run SQL*Plus statements stored in a file	`@ { url | filename [ .ext ] } [ arg... ]`
	`START filename [ .ext ] [ arg... ]`
	`.ext can be omitted if the filename extension is .sql`

Execute commands stored in the SQL buffer	`/`
List and execute commands stored in the SQL buffer	`RUN`
Execute a single PL/SQL statement or run a stored procedure	`EXECUTE statement`
Disconnect from a database	`DISCONNECT`
Shut down a database	`SHUTDOWN [ ABORT | IMMEDIATE | NORMAL | ... ]`

Log out of SQL*Plus	`{ EXIT | QUIT }`
	`[ SUCCESS | FAILURE | WARNING | ... ]`
	`[ COMMIT | ROLLBACK ]`

Login to Oracle Server

You can use SQL Client to connect to Oracle Server (Dbank), the following credential can be used to connect to DBServer.

User name: KxxYYYY e.g k132000

Password: fast

Host string: Dbank

Changing Password for your user

Write the following command to SQL prompt:

SQL > alter user kxxYYYY identified by “123456”;

This command is used to alter user information. Changing password is by identification change.

SQL> select user from dual;

Viewing your Objects

Write the following command to SQL prompt:

SQL > select * from cat;

This command is used to retrieve category objects from a user. You will find no-row in this case. As you have not created any object in your account so far.

Schema

In oracle an example database is Scott schema, it contains the 5 tables.

Run this demobld.sql from SQL Client.

SQL> @c:\demobld.sql

Viewing your Objects

Write the following command to SQL prompt:

SQL > select * from cat;

Now you will find four rows in this case. As you have created objects in your account.

OUTPUT:

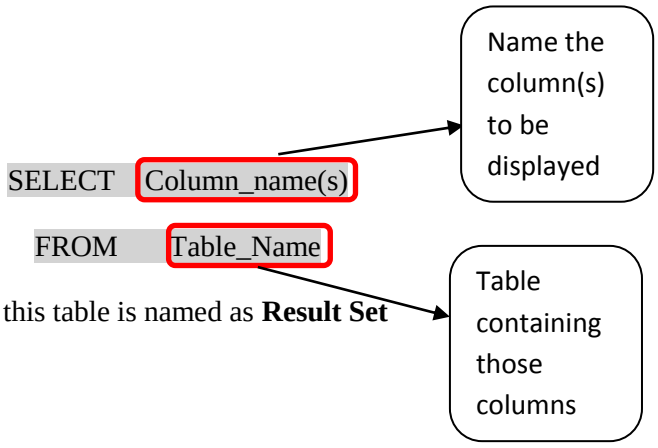
TABLE_NAME	TABLE_TYPE
BONUS	TABLE
DEPT	TABLE
EMP	TABLE
SALGRADE	TABLE

RETRIEVING DATA USING THE SQL SELECT STATEMENT

The SELECT statement is a DML (Data Manipulation Language) statement. DML statements are SQL commands that allows you to retrieve and manipulate a data in the database.

Basic Select Statement

Syntax:



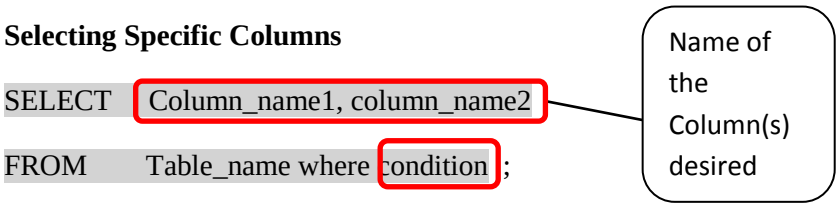
The result is stored in a result table; this table is named as **Result Set**

Selecting All Columns

```
SELECT * FROM TABLE_NAME
```

Here '*' (asterisk) means **'All'**

Selecting Specific Columns



Example:

- ✓ **Display all records from EMPLOYEE Table**

```
SELECT * FROM EMP;
```

// '*' denote all attributes. See other tables.

- ✓ **Display DEPARTMENT NUMBER, EMPLOYEE NUMBER, JOB From EMPLOYEE Table**

```
SELECT DEPTNO,EMPNO,JOB FROM EMP;
```

- ✓ **Display DEPARTMENT NO, EMPLOYEE NO, JOB, SALARY From EMPLOYEE Table**

```
SELECT DEPTNO,EMPNO,JOB,SAL FROM EMP;
```

- ✓ **Show employees information whose Salary is greater than 2000**

```
SELECT * FROM EMP WHERE SAL>2000;
```

- ✓ **To count No of Rows in a Table using COUNT()**

```
SELECT COUNT(*) FROM EMP;
```

- ✓ **Show employees information whose Salary is greater than 2000**

```
SELECT * FROM EMP WHERE SAL>2000;
```

- ✓ **Display The Current Date. Label The Column Date**

```
SELECT  sysdate "Date" FROM    dual;
```

Using Arithmetic Operators

```
SELECT 2*6
FROM dual;
```

Using Date Arithmetic

```
Select TO_DATE ('31-jul-2012') + 2
FROM DUAL;
```

To-Date () is a function that converts a string to a Date.

```
Select TO_DATE ('02-AUG-2012') – TO_DATE ('31-JUL-2012')
FROM dual;
```

IMPORTANT

Dual is a built-in table that contains a single row has one varchar2 column named Dummy. You can use the dual table to perform simple queries

---

Using Column aliases

A column Alias:

Renames a **column heading** and uses an *optional* keyword '**AS**' between the column name and alias. It requires double quotation ("") mark if the alias contains spaces or special characters or if it is case-sensitive.

Syntax:

```
Select column_name1 AS Column_Alias1, Column_name2 Column_Alias2 FROM
TABLE_NAME;
```

Optional

Column Headings (Aliases)

Concatenation Operator

A Concatenation Operator:

Links columns or character strings to other columns and is represented by two vertical bar **(||).** It creates a resultant column that is a character expression

```
SELECT Column_name1 || ‘  ‘ || Column_name2 AS “ANY ALIAS”
FROM TABLE_NAME;
```

Space character is concatenated to the First Column and the resulting string is concatenated to the Second Column.

- ✓ **Display Employee Name and JOB**

```
SELECT ename||’, ’||job  "Employee and Title" FROM   emp;
```

ARITHMETIC EXPRESSIONS

Create expressions with number and date data by using arithmetic operators.

Operator	Description
+	**Add**
-	**Subtract**
*	**Multiply**
/	**Divide**

Operator Precedence

```
Select 10 * 12 / 3 – 1
FROM DUAL;
```

Result will be?

Applying parenthesis:

```
SELECT 10 * ( 12 / 3 -1)
FROM DUAL;
```

Displaying Distinct Rows

SELECT DISTINCT COLUMN_NAME FROM TABLE_NAME**;**

In order to refrain from duplicate records, DISTINCT keyword is used

- ✓ **Show names of all jobs in a department.**
  SELECT DISTINCT JOB FROM EMP ;
- ✓ **Display no of jobs in a department**

  SELECT COUNT(DISTINCT JOB) FROM EMP;

- ✓ **List the employees who joined in the year 81.**

  SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,'YY')='81';

- ✓ **List the emps who are joined in the month of Aug 1980.**

  SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,'MON-YY')='AUG-80';

USING COMPARISION OPERATORS;

Operator	**Description**
=	**Equal**
<> or !=	**Not Equal**
<	**Less than**
>	**Greater than**
<=	**Less than or equal to**
>=	**Greater than or Equal to**

- SELECT * FROM emp WHERE deptno <> 10;

Here all conditions must be true in order to evaluate the whole expression as True

USING SQL OPERATORS

LIKE	**Matches Patterns In Strings**
IN	**Matches Lists Of Values**
BETWEEN	**Matches A Range Of Values**
IS NULL	**Matches Null Values**

Using Like Operator

Like operator is used in pattern matching

- **Underscore Character (_)** Matches one character in a specific position
- **Percent Character (%)** Matches any number of characters beginning at the specified position

Syntax:

SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern

E.g.

SELECT *

FROM EMP

WHERE ename LIKE '_O%'

2nd character must be O

Only one character

Any number of characters at the End of the string

SELECT * FROM EMP WHERE ename LIKE **'%J%'**

- ✓ **Display the names of all employees with names starting with S.**

SELECT ENAME FROM EMP WHERE ENAME LIKE 'S%';

- ✓ **Display the names of all employees with second character of name as A.**

SELECT ENAME FROM EMP WHERE ENAME LIKE '_A%';

- ✓ **List the employee names starting with 'K' and ending with 'S'.**

SELECT ENAME FROM EMP WHERE ENAME LIKE 'K%S'

Similarly, use NOT LIKE keyword to reverse the rows retrieved by previous query.

- ✓ **List the employees whose Employee number not starting with digit78.**

  SELECT * FROM EMP WHERE EMPNO NOT LIKE '78%';

To Display Certain No Of Rows:

SELECT column_name(s)

FROM table_name

WHERE ROWNUM <= number

No of Rows to be returned

- ✓ SELECT EMPNO, ENAME, DEPTNO FROM EMP WHERE ROWNUM <= 3;

Using IN Operator

It is used to select only those rows whose column value is in a list that you specify.

Syntax:

SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1,value2,...)

- ✓ **List the employees who are working for the Department number 10 or20.**

  SELECT * FROM EMP WHERE DEPTNO IN (10,20);

  Similarly, use NOT IN keyword for the reverse of this query Output

- ✓ **List all the emps except 'PRESIDENT' & 'MGR"**

  select * FROM EMP WHERE JOB NOT IN ('PRESIDENT','MANAGER');

Using BETWEEN operator

The BETWEEN operator selects a range of data between two values. The values can be numbers, text, or dates.

SELECT column_name(s)
FROM table_name
WHERE column_name
BETWEEN value1 AND value2

Range of record(s) to be retrieved

- ✓ **List the emps Who Annual sal ranging from 30000 and 50000.**

  SELECT * FROM EMP WHERE SAL*12 BETWEEN 30000 AND 50000;

Defining NULL Values

A null is a value that is unavailable, unassigned, unknown, or inapplicable. It is not possible to compare null values and zero. They are not equivalent.

- ✓ **List the emp_no and name of employee who has no manager.**

  SELECT empno, ename FROM emp WHERE mgr IS NULL;

How can we Replace Null Values?

SELECT NVL (SomeNullableField, 'If null, this value') → Value to be replaced with

FROM TABLE_NAME;

- ✓ **Displays the employees' names and commission amounts. If an Employee does not earn commission, show "no commission." Label the column COMMISION LIST**
  SELECT ename, NVL(TO_CHAR(COMM), 'No Commission') "COMMISSION LIST" FROM emp;

Sorting Rows Using the ORDER BY Clause

ORDER BY clause to sort the rows retrieved from database.

SELECT * FROM Table_name ORDER BY Column_name(s)

Similarly, We can sort data in Ascending and Descending Order

SELECT *

FROM EMP

ORDER BY ENAME DESC;

DESC FOR DESCENDING ORDER

ASC FOR ASCENDING ORDER

ASC is the default order.

- ✓ **Display all the unique job groups in the descending order?**

  SELECT DISTINCT JOB FROM EMP ORDER BY JOB DESC;

- ✓ **List the employees who joined on 1-MAY-81,3-DEC-81,17-DEC-81,19-JAN-80 in asc order of seniority.**
  SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,'DD-MON-YY') IN ('1-MAY-81','3-DEC-81''17-DEC-81''19-JAN-80') ORDER BY HIREDATE;

ACTIVITY

1. Display the name, job, and salary for all employees whose job is Clerk or Analyst and their salary are not equal to Rs.1000, Rs.3000, or Rs.5000. Display in descending order of salary.
2. Display "Employee Joining Info" containing Employee name and their joining date for each employee in the EMPLOYEE table.
3. List all the employees names, their department numbers and hire date with ascending order of dept numbers.
4. Find emp.no, name, salary and hire date of the employees who were hired in the first half year of 1981. Sort the output in ascending order of hiredate.(5 rows)
5. List name of employees along with *name* of their departments. Sort by department name.
6. How many different job titles are stored in the relation emp
7. How many employees earn more than Rs. 2000.
8. Find name and job of employees whose name contain substring 'LL'.
9. Selecting employee number, name and their salary who do not earn Commission.
10. List the employees in the ascending order of their salaries.
11. List the Employee no, Employee name, Salary of all managers.
12. Write a query which concat the job and salary of a employee working in sales department.
13. List all employees whose name contains A in the third position.

BEST OF LUCK