Database Management Systems

Laboratory Manual

G. Mahesh

Associate Professor

(For private circulation among the students)

Department of Computer Science and Engineering

S.R.K.R. Engineering College

Bhimavaram – 534204.

Dedicated to

To Honorable Secreatary and Correspondent of our College
Sri S.Atchutha Rama Raju

To Our beloved Head of C.S.E. Department
Prof. G.V.Padma Raju

Author:
G.MAHESH,
Associate professor,
Dept of C.S.E.

Co-Authors:
K.V Krishnam Raju,
Associate professor,
Dept of C.S.E.

K.Aruna Kumari,
Assistant professor,
Dept of C.S.E.

Index

Expno: 01 **Date:**

Simple SQL commands

Aim:-

To Practice simple SQL commands for retrieving data contained in **'Emp'** and **'Dept'** tables.

Theory:-

Questions including data stored in a Data Base are called Queries. The SQL stands for structured query language. It is a non procedural language used to define, access and manipulate data. The meaning of non-procedural is that it describes the necessary components (i.e., tables) and desired results without dictating exactly how results should be computed.

To bring conformity among vendors, the American National Standards Institute (ANSI) published its first SQL standard in 1986 and a second widely adopted standard in1989. ANSI released updates in 1992, known as SQL92 and SQL2, and again in 1999, termed as both SQL99 and SQL3.

In SQL92, SQL statements are grouped into three broad categories. They are the Data Manipulation Language (DML), the Data Definition Language (DDL), and the Data Control Language (DCL). The DML provides specific data-manipulation commands such as SELECT, INSERT, UPDATE, and DELETE. The DDL contains commands that handle the accessibility and manipulation of database objects, including CREATE and DROP, while the DCL contains the permission-related commands GRANT and REVOKE.

The SELECT statement is used to retrieve data from a database. The basic form of an SQL query is

select [**distinct**] select-list **from** from-list **where** qualification;

Where select, distinct, from and where are key words. Select-list is list of column names separated by commas and from-list is table or list of tables from which data is obtained. If from-list is list of tables, then the result is cross product of both table's contents. The Qualification in the optional where clause is a Boolean combination of conditions. The optional distinct key word is used to select unrepeated values.

We can also use an optional order by clause to display the contents in sorting order.

Relational operators are <, <=, >, >=, <>, =

They are for less than, less than or equal to, greater than, greater than or equal to, not equal to, equal to.

Logical operators are **and, or** and **not.**

Some rules about SQL statements are

- SQL statements are not case sensitive but values like scott, SCOTT are different.
- SQL statements can be on one or more lines.
- Keywords cannot be abbreviated or split across lines.
- Statement ends with a semicolon.

Procedure:

1. For getting SQL prompt
   a. Click start
   b. Click all programs
   c. Click Oracle
   d. Click Applicattion development
   e. Click SQL plus
2. At Logon window enter the following data.
   a. User name : system
   b. Password : manager
   c. Host string : oracle9
3. After getting SQL prompt type the following command to create a new user user001 for dialy log in with 100mb of space.

   create user user001
   identified by password001
   quota 100m **on users**;
4. Grant permissions to user using the following command.

   grant connect,create session,resource to user001**;**
5. Now copy emp and dept tables from scott to user001 using the folowing commands,

   create table user001.dept **as select * from** scott.dept**;**
   create table user001.emp **as select * from** scott.emp**;**
6. Now use the following sequence to save the work and quit from SQL prompt.

   commit;
   exit;
7. Again go to SQL prompt.

8. At Logon window enter the following data.
   a. User name : user001
   b. Password : password001
   c. Host string : oracle9
9. Now type the foolowing commands start the work.
   show user;
   set linesize 100**;**
10. Now start doing the queries and note down the queries and results.
    It may be noted that after executing every SQL command we have to execute
    select * from table;
    and check manually that the result is the desired one or not.

Questions:-

1. Display the contents of the emp table.

   Ans. (SQl query): Select * from emp;
2. Display the contents of the dept table.
3. Display all the employees' names.
4. Display all the department names.
5. Display all the employees' names along with their designations.
6. Display the details of employees whose salaries are greater than 2000.
7. Display the details of employee whose empno is 7788.
8. Display the designation of employee whose empno is 7788.
9. Display the name, designation and salary of employees whose salaries are greater than 2000.
10. Display the name, designation and salary of employees whose salaries are less than 2000.
11. Display the name, designation and salary of employees whose salaries are between 2500 and 3000.
12. Display the name, designation and salary of employees whose salaries are less than 2500 and that of employees whose salaries are greater than 3000.
13. Display the details of employee whose name is scott.

14. Display the details of employees whose designation is manager.
15. Display the names and designation of employees whose designation is manager and whose salary is greater than 2500.
16. Display the details of employees whose names start with letter s.
17. Display the details of employees whose names end with letter t.
18. Display the salaries of employees without repetitions.
19. Display names of employees without any commission.
20. Display names of employees with some commission along with commission.
21. Display names of employees and salaries if the salaries were incremented by 200.
22. Display names and designation of employees after concatenating.
23. Display names of employees in an alphabetical order.
24. Display empno of employees and salaries in order of salaries such that employee with max salary is on top.
25. Display names of employees and salaries in order of salaries such that employee with max salary is at the bottom.
26. Display names of all employees with the heading as employee_name.
27. Display names of employees and joining dates in the order of joining dates.
28. Display the details of salesman and manager whose salary is 1500 or more.
29. Calculate and display salary of scott for one year.
30. Display location of department number 20.
31. Display department number of scott.
32. Display location of the department in which ‘scott’ is working.

Expno: 02 **Date:**

Creation of tables

Aim:-

To create, modify and update tables and to practice some aggregate functions.

Theory:-

Create table: Tables are defined using the create command. The simplified form of create command is

```
create table  r
( A1  D1, A2  D2,..., An  Dn,
[ integrity_constraint1 ],...,[ integrity_constraintn])
| as select_statement;
```

Where create and table are key words.

- **create, table** and **as** are keywords.
- r is the name of relation
- A1, A2, .. , An are attributes of the relation separated by domain types D1,D2,...
- the way of mentioning integrity constraints are
  1. **primary key**(attribute).
     It is a set of attributes that uniquely identifies a tuple in a relation and selected by the DBA for the purpose.
  2. **foreign key** (Ar) **references** r1.
     or
  3. **constraint** c1 **foreign key** (Ar) **references** r1.
     Here c1 is constraint name. Foreign key is a combination of attributes with values based on the primary key values from another table. The meaning of this referential integrity here is that for each tupple in the relation, the values of 'Ar' must exist in the referring relation.
  4. **check**(condition)
- The optional select statement can be used to create a table from another reference table.

insert : Tables are constructed using the insert command. The simplified form of insert command is

```
insert into r
[(A1, A2, ..,An)] values(v1, v2, ..,vn)
| select statement ;
```

- **insert**, **into** and **values** are key words.
- r is the name of relation
- A1, A2, … ,An are attributes of the relation and v1,v2,.., vn are the corresponding values. If the optional attribute list is not given, all the attributes in the relation are assumed.
- The optional select statement can be used to insert values from another table. But here it is to be ensured that both tables possess compatible data types.

update: The attributes values are updated using the update command. The simplified form of update command is

```
update table_name | view_name
set A = v1
where conditions;
```

- **update**, **set** and **where** are key words
- A is attribute whose new value will be v1.
- The attribute values for the all rows satisfying the conditions are updated.
- If where condition is not given, the attribute values for the all rows are updated.

drop table: Table can be dropped using drop table. The syntax of the command is

```
drop table [owner.] table_name;
```

- **drop table** is a key word.

Dropping a table in Oracle frees the space used by the table and commits any pending changes to the database. All indexes and grants associated with the table are lost. Objects, such as views, stored procedures, and synonyms built upon the table, are marked invalid and cease to function.

Delete: Table contents can be deleted using delete command. The syntax of delete command is

```
delete from [owner.] table_name [where clause];
```

- **Delete**, **from** & **where** are key words
- The attribute values for the all rows satisfying the conditions are deleted.
- If where condition is not given, the attribute values for the all rows are deleted.

Alter table: Table structure can be altered using alter table command. The syntax of alter table command is

```
alter table [owner_name.] table_name
 add A D
| modify A D
| drop column A ;
```

- **alter table, add , modify, drop**, **column** are key words
- **add** , **modify**, **drop** are used to append an attribute, to change the data type of an existing attribute, and to delete an attribute respectively.
- A is attribute and D is domain type.

Aggregate functions:

Aggregate functions are functions that take a collection of values as input and return a single value. SQL offers five built-in aggregate functions.

- **MIN** (A) :returns minimum of column values
- **MAX** (A) :returns maximum of column values
- **AVG** ([**distinct**] A) :returns average of unique column values
- **SUM** ([**distinct**] A) :returns total of unique column values
- **COUNT**([**distinct**]A):returns number of unique column values

The input to **SUM** and **AVG** must be a collection of numbers but the other operators can operate on collection of non-numeric data types like strings.

For example,

select sum(sal) **from** emp;	displays total salaries of all the employees.
select count(com) **from** emp;	displays no. of employees with commission.
select count (*) **from** emp;	displays no. of records in emp.

All the functions except COUNT (*) ignore null values in their input collection.

For SQL null value means unknown or inapplicable.

group by & having Clauses :

The select statement syntax including optional group by and Having clauses is as under :

select [**distinct**] select-list **from** from-list

[**where** qualification]

[**group by** group-list]

[**having** group-qualification];

The result of the GROUP BY clause is a grouped table i.e., a set of groups of rows derived from the table by conceptually rearranging it into the minimum number of groups such that within any one group all rows have the same value for the combination of columns identified by GROUP BY clause.

Ex : **select sum**(sal) from emp **group by** job;

Using this statement we get total salary for each designation of the employees.

There is a restriction on the select clause while using group by. Select item in the select clause must be single valued per group as for the other aggregate functions. Therefore a statement like

select empno, sum(sal) **from** emp **group by** job;

select * **from** emp **group by** job; are invalid.

But we can give

select job **from** emp **group by** job;

and

select job, avg(sal) **from** emp **group by** job;

are valid.

HAVING clause works very much like a where clause except that its logic is related only to the result of group functions i.e., having clause is used to eliminate groups where as where clause is used to eliminate rows from the whole table.

For example,

select deptno, avg(sal) from emp group by deptno **having** max(sal) > 1500;

Displays average salaries depart wise for departments which have average salary greater than 1500.

Procedure:-

1. Go to SQL prompt
    a. Click start
    b. Click all programs
    c. Click Oracle
    d. Click Application development
    e. Click SQL plus
2. At Logon window enter the following data.
    f. User name : user001
    g. Password : password001
    h. Host string : oracle9
3. Go to File menu, click spool and then spool file.
4. Give the spool file name like **atoz**
5. Click **save**
6. Set the line size using the following command.
    set linesize 100;
7. Create the tables with create table command.
8. Type the following command after creation or modification of each table to view the structure of the table.
    desc table_name;
9. In **insert** statement we can use substitution variables like
    insert into table1 **values**(&a, &b, &c);
10. Then it asks for the values of a, b, and c. If we give proper values, the values are inserted into the table.
11. For inserting more rows of data we can repeat the command by typing ‘ /’.
12. In between insertions and after all the data is inserted, we can type the following command to save the data.
    commit;
13. To cancel the changes to the database up to the last save point use
    rollback;
14. After inserting all the data or after modification to data, we can view the data by typing
    Select * from table_name;
15. Use the following commands after all the task is completed.
    commit;
    exit;

16. Open the spool file **atoz** with **notepad** and note down the contents in the observation.

Questions:-

1. Create emp1 table and insert appropriate values.
2. Create dept1 table and insert appropriate values.
3. Create suppliers table with the following schema
   suppliers(sid:integer,sname:string,address:string).
4. Create parts table with the following schema
   parts(pid:integer, pname:string, color:string).
5. Create catalog table with the following schema
   catalog(sid:integer , pid:integer , cost:real).

6. Insert the following values into suppliers table

SID	SNAME	ADDRESS
1001	smith	london
1002	jones	paris
1003	blake	paris
1004	clark	london
1005	adams	Athens

7. Insert the following values into catalog table

SID	PID	COST
1001	5001	10.5
1002	5003	50
1002	5005	550.99
1003	5004	49.55
1004	5006	600
1005	5002	200
1005	5005	545
1005	5006	605
1005	5001	10
1005	5003	49
1005	5004	50

8. Insert the following values into parts table

PID	PNAME	COLOR
5001	nut	red
5002	bolt	green
5003	screw	blue
5004	screw	red
5005	cam	blue
5006	cog	red

9. Increment the salaries of all employees by 500 except scott for whom increment the salary by 1000.
10. Add one row of your own choice and delete that row in parts table.
11. Alter the ename field of emp1 table to accommodate more characters.
12. Find the maximum salary.
13. Display maximum salary along with name of the employee who gets it.
14. Display total salary.
15. Display total salary for each department.
16. Display the maximum salary of employees in each department along with the department number
17. Count number of records in the emp1 table.
18. Count number of employees who get a commission.
19. Count the number of employees who get no commission.
20. Create a table dummy with the fields serial number, name and insert 5 rows.
21. Delete all the rows in dummy table.
22. Delete the field “name” from dummy table.
23. Create table emp2 with same data as in emp.
24. Drop table dummy.

Note: the schema for tables emp1 and emp2 is same as the emp.dept tables of scott. So before creating these tables , we have to note down the schema of emp and dept tables by the desc command.

Expno: 03 **Date:**

Set Operations

Aim:-

To Practice set operations, nested queries and joins in SQL

Theory:-

A nested query is query that has another query embedded within it, the embedded query is called sub query.

Example:

```
select dname from dept  where deptno
        = (select deptno from emp where ename='scott');
```

This query displays department name of the employee whose name is 'scott'. If a single query is used on emp table then we can obtain deptno but not dname. So we used the result of the sub query in the nested query.

There are some points to be remembered while using nested queries. They are

- Use single-row operators with single-row sub queries and use multiple-row operators with multiple-row sub queries.
- In Single row sub query, operators used are <, >, <= , >=, <>
- In multiple row sub query, operators used are **all**, **some** and **in**. whose meanings are every value returned by the sub query, at least one value returned by the sub query , any value in the list respectively.

SQL supports the set operations union, intersection and minus with the key words union, intersect, and except (minus in oracle). It also supports exists and not exists. exists operator returns true if the set is not empty and not exists returns true if the set is empty.

The UNION operator returns records from the result of both queries after eliminating the duplicate records which are common in both. There is another option of union, the UNION ALL operator, which returns records from the result of both queries, including duplicate records which are common in both. The INTERSECT operator returns the records which are common in the result of both queries. The EXCEPT operator returns the records which are in the result of the first query but not in the result of the second one.

We can use the join capability in SQL to bring together data that is stored in different tables by creating a link between them.

When data from more than one table in the database is required, a join condition is used. Rows in one table can be joined to rows in another table according to common values existing in corresponding columns, that is, usually primary and foreign key columns. A join with a join condition is known as conditional join.

When a join condition is invalid or omitted completely, the result is a Cartesian product, in which all combinations of rows are displayed. All rows in the first table are joined to all rows in the second table.

Equi-Join is a special case of condition join where the condition contains only equalities. A natural join is an Equi-Join on all common fields of the related tables.

If there are any values in one table that do not have corresponding values in the other, in an Equi-join that row will not be selected. Such rows can be forcefully selected by using outer join. The left outer join takes all the rows in the left table (first table in the join) and pads the rows which don't have a matching row with null values for all other attributes in the right table. The right outer join takes all the rows in the right table(first table in the join) and pads the rows which doesn't have a matching row with null values for all other attributes in the left table. The full outer join includes all the rows from both tables and pads the rows which doesn't have a matching row with null values for all other attributes in the other table.

In SQL there are four join types namely inner join, left outer join, right outer join and full outer join. And there are three join conditions natural, on <condition> , using (field1,field2,..).Here on <condition> is for conditional join and using (field1,field2,..) is like natural join but the join fields are fields specified in the using class.

Examples:

Cartesian product:

The following statement displays combinations of all records in dept and emp tables.

select * **from** dept,emp;

conditional joins:

The following statement applies Cartesian product on dept and emp tables and displays the records which satisfy the condition i.e. the details of department no 10 employees' details who doesn't belong to the department.

select * **from** dept,emp **where** dept.deptno<>emp.deptno **and** dept.deptno=10;

Equi-join:

The following statement applies Cartesian product on dept and emp tables and displays the records which satisfy the condition i.e. the details of departments which have some employees along with employees' details who belong to the department. The condition here is equality condition. So, it is also known as Equi-join.

select * from dept,emp **where** dept.deptno=emp.deptno;

inner join:

An alternative way of using the above query is

select * **from** dept **inner join** emp **on** dept.deptno=emp.deptno;

left outer join:

The following statements display all the dept records which have the matching emp records along with the matching emp records and in addition displays dept records which does not have any employees.

select * **from** dept **left outer join** emp **on** dept.deptno=emp.deptno;

OR

select * **from** dept, emp **where** dept.deptno =emp.deptno(+);

right outer join:

The following statements display all the dept records which have the matching emp records along with the matching emp records and in addition displays emp records which does not have any department.

select * **from** dept **right outer join** emp **on** dept.deptno=emp.deptno;

OR

select * **from** dept, emp **where** dept.deptno(+) =emp.deptno;

full outer join:

The following statements display all the dept records which have the matching emp records along with the matching emp records and in addition displays emp records which does not have any department and dept records which does not have any employees.

select * **from** dept **full outer join** emp **on** dept.deptno=emp.deptno;

natural inner join:

The following statement inner joins the two tables by equating all the common fields i.e. in this case it is deptno. So the result is same as the above inner join but in the display the common column deptno is displayed only once.

select * **from** dept natural **inner join** emp;

natural left outer join:

The following statement left outer joins the two tables by equating all the common fields i.e. in this case it is deptno. So the result is same as the above left outer join but in the display the common column deptno is displayed only once.

select * **from** dept **natural left outer join** emp;

natural right outer join:

The following statement left outer joins the two tables by equating all the common fields i.e. in this case it is deptno. So the result is same as the above right outer join but in the display the common column deptno is displayed only once.

select * **from** dept **natural right outer join** emp**;**

natural full outer join:

The following statement full outer joins the two tables by equating all the common fields i.e. in this case it is deptno. So the result is same as the previous full outer join but in the display the common column deptno is displayed only once

select * **from** dept **natural full outer join** emp**;**

Procedure:-

Same as that in expno.2

Questions:-

1. Create three tables with the following schema

   Sailors(sid:integer, sname:string, rating:integer, age:real)

   Boats(bid:integer, bname:string, color:string)

   Reserves(sid:integer, bid:integer, day:date)

   Give the appropriate integrity constraints.

2. Insert the following values into the above tables.

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	40
85	Art	3	25.5
95	Bob	3	63.5

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

3. Find the names of sailors who have reserved at least one boat.
4. Find the ids of sailors who reserved a red boat and a green boat.
5. Find the ids of sailors who reserved a red boat or a green boat.
6. Find the ids of sailors who reserved a red boat but not a green boat.
7. Find the names of sailors who reserved every boat.
8. Find the names of sailors who reserved every red boat.
9. Find the ids of sailors whose rating is more than the average rating of all sailors.
10. Find the names of parts for which there is some supplier.
11. Find the ids of suppliers who supply a red part and a green part.
12. Find the ids of suppliers who supply a red part or a green part.
13. Find the ids of suppliers who supply a red part but not a green part.
14. Find the names of suppliers who supply every part.
15. Find the names of suppliers who supply every red part.
16. Find the names of parts supplied by Adams and by no one else.
17. Find the ids of suppliers who charge more for some part than the average cost of that part.
18. For each part find the name of the supplier who charges the most for that part.
19. Find the ids of suppliers who supply only red parts.
20. Find the sids of sailors whose rating is better than some sailors called horatio.
21. Find the sailors whose rating is better than every sailor called horatio.
22. Find the names of boats reserved by dustin and by no one else.

Expno: 04 **Date:**

Functions and Procedures

Aim:-

To create and execute functions and procedures.

Theory:-

PL/SQL stands for Procedural Language/Structured Query Language.PL/SQL is the procedural extension to SQL with design features of programming languages. Data manipulation and query statements of SQL are included within procedural units of code. PL/SQL Block Structure is as follows.

```
<header>
IS | AS
    Declaration section
BEGIN
    Executable section
EXCEPTION            (optional)
    Exception section
END;
```

Section	Description
Header	Required for named blocks. Specifies the way the program is called by other PL/SQL blocks. Anonymous blocks do not have a header. They start with the DECLARE keyword if there is a declaration section, or with the BEGIN keyword if there are no declarations.
Declaration	It is optional. Declares variables, cursors, TYPEs, and local programs that are used in the block's execution and exception sections.
Execution	Optional in package and TYPE specifications; contains statements that are executed when the block is run.
Exception	Optional; describes error-handling behavior for exceptions raised in the executable section.

An example of an anonymous block:

The following code declares today as date data type and the default value is system date which can be obtained from the environment variable SYSDATE. Display the date after concatenating it with the string.

```
DECLARE
  today DATE DEFAULT SYSDATE;
BEGIN
  DBMS_OUTPUT.PUT_LINE ('Today is ' || today);
END;
```

Syntax of the named blocks (procedures and functions):

```
CREATE [OR REPLACE] PROCEDURE [owner_name.]procedure_name
[(parameter1 [IN | OUT | IN OUT] datatype][,...n)]]
{IS | AS} PL/SQL block ;
```

```
CREATE [OR REPLACE] FUNCTION [owner_name.]function_name
[(parameter1 [IN | OUT | IN OUT] datatype][,...n)]]
RETURN datatype
{IS | AS} PL/SQL block;
```

In Oracle, user-defined functions and stored procedures are very similar in composition and structure. The primary difference is that stored procedures cannot return a value to the invoking process, while a function may return a single value to the invoking process. In an Oracle stored procedure, the specified arguments and parameters include IN, OUT, or IN OUT. The IN qualifier is provided when invoking the function and passes a value in to the function, while OUT arguments pass a value back to the invoking process. In other words, the IN qualifier is supplied by the user or process that calls the function, while the OUT argument is returned by the function. IN OUT arguments perform both IN and OUT functionality.

Example:

Procedure that computes and displays tax to be paid if income is passed as parameter

```
CREATE OR REPLACE PROCEDURE tax1(p_value IN NUMBER)
IS
BEGIN
  DBMS_OUTPUT.PUT_LINE (p_value * 0.08);
END;
```

Example:

Function that returns tax to be paid if income is passed as parameter

```
CREATE OR REPLACE FUNCTION tax2(p_value IN NUMBER)
RETURN NUMBER
IS
BEGIN
  RETURN (p_value * 0.08);
END;
```

Control statements in PL/SQL

If Statement:

IF *condition* THEN
executable statement(s)
END IF;

If – else Statement

IF *condition* **THEN**
TRUE sequence_of_executable_statement(s)
ELSE
FALSE/NULL sequence_of_executable_statement(s)
END IF;

elsif Statement:

IF *condition-1* **THEN**
statements-1
ELSIF condition-N **THEN**
statements-N
[**ELSE**
ELSE statements]
END IF;

loop Statement:

```
LOOP
  executable_statement(s)
END LOOP;
```

exit Statement:

```
EXIT [WHEN condition];
--
FOR loop_index IN [REVERSE]
lowest_number..highest_number
LOOP
  executable_statement(s)
END LOOP;
```

The REVERSE keyword causes PL/SQL to start with the highest_number and decrement down to the lowest_number

while Statement:

```
WHILE condition
LOOP
  executable_statement(s)
END LOOP;
```

case Statement:

```
CASE selector
  WHEN expression1 THEN result1
  WHEN expression2 THEN result2
  ...
  WHEN expressionN THEN resultN
 [ELSE resultN+1;]
END;
```

Data types:

- **CHAR** [(*maximum_length*)]
- **VARCHAR2** (*maximum_length*)
- **NUMBER** [(*precision, scale*)]
- **DATE**
- **%TYPE** Attribute

Declares a variable according to a database column definition or previously declared variable

Example:

Procedure that displays name of an employee if his empno is given. It uses the %type.

```
CREATE OR REPLACE PROCEDURE pemp1(no emp.empno%type)
IS
empname emp.ename%type;
BEGIN
  SELECT   ename
  INTO     empname
  FROM     emp
  WHERE    empno = no;
DBMS_OUTPUT.PUT_LINE (empname);
END;
/
```

Cursors:

the syntax of cursor declaration is

DECLARE CURSOR cursor_name

IS select_statement

[**FOR UPDATE** [**OF** column_name [,...n]]}]

The DECLARE CURSOR command enables the retrieval and manipulation of records from a table one row at a time. This provides row-by-row processing, rather than the traditional set processing offered by SQL. To use this procedure properly, you should:

1. **DECLARE** the cursor
2. **OPEN** the cursor
3. **FETCH** rows from the cursor
4. When finished, **CLOSE** the cursor

The DECLARE CURSOR command works by specifying a SELECT statement. Each row returned by the SELECT statement may be individually retrieved and manipulated. In Oracle, variables are not allowed in the WHERE clause of the SELECT statement unless they are first declared as variables. The parameters are not assigned at the DECLARE. Instead, they are assigned values at the OPEN command.

Example:

The following code creates procedure to display empno and ename of all employees in emp table using cursors.

```
CREATE OR REPLACE PROCEDURE pemp2
IS
        CURSOR c IS
        SELECT empno, ename
        FROM   emp;
        no emp.empno%type;
        name emp.ename%type;
BEGIN
        OPEN c;
  LOOP
        FETCH c into no, name;
        EXIT WHEN c%notfound;
        DBMS_OUTPUT.PUT_LINE (TO_CHAR(no)||'-->'|| name);
  END LOOP;
CLOSE c;
END;
/
```

Example:

The following code creates procedure to update salaries of employees using cursors

```
CREATE OR REPLACE PROCEDURE p2
IS
        CURSOR c1 IS
        SELECT deptno,empno,ename,sal
        FROM   emp FOR UPDATE OF sal;
BEGIN
        FOR emp_record IN c1
        LOOP
                IF emp_record.sal> 2000 THEN
                UPDATE emp
                SET    sal=emp_record.sal* 1.10
                WHERE CURRENT OF c1;
                END IF;
        END LOOP;
 END;
```

Procedure:-

1. Go to SQL prompt
2. Execute the following commands.

   set autocommit on;

   set serveroutput on;
3. Type edit file1.sql (Then the default editor i.e. notepad is invoked.)
4. Type the procedure or function definition.
5. Save it and exit notepad.
6. Compile the procedure or function by typing the following command.

   @file1
7. If there are any errors type the following command to show the errors.

   Show errors;
8. Type the following command to edit the file and introduce the corrections.

   edit file1.sql
9. Save the file and exit the editor.
10. Compile the procedure by typing the following command again.

    @file1
11. If there are any errors type the following command to show the errors.

    show errors;
12. Continue the process of reediting and recompilation until the procedure or function is successfully created.
13. After successful creation of procedure type the following command to execute the procedure.

    exec procedurename(parameter);

    example1 **exec** pemp1;

    example2 **exec** tax1(1000);
14. If it is a function type the following command to execute the function.
    a. **select** tax2(1000) **from** dual;

       Or

    b. Execute Following sequence of statements for getting the value returned by the function into bind variable and print it.

       1. **variable** v1 **number**;
       2. **exec :**v1**:=**tax2(1000)**;**
       3. **print** v1;

15. We can view the function names and procedure names using the command.

```
select object_name, object_type
from user_objects
where object_type in ('PROCEDURE','FUNCTION')
order by object_name;
```

16. We can view the function or procedure body using the command.

```
select text
from user_source
where name ='TAX1' ;
```

17. Exit SQL after saving the work.

```
commit;
exit;
```

Expno: 05 **Date:**

Triggers

Aim:-

The aim of the experiment is to create views and triggers and to study the behavior of views and triggers.

Theory:-

A trigger is a special kind of stored procedure that fires automatically (hence, the term trigger) when a data-modification statement is executed. Triggers are associated with a specific data-modification statement (INSERT, UPDATE, or DELETE) on a specific table.

Syntax and Description:

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF}
{[DELETE] [OR] [INSERT] [OR] [UPDATE [OF column [,...n] ]] ON
{table_name | view_name}
[FOR EACH { ROW | STATEMENT }]
[WHEN (conditions)]
code block
```

Triggers, by default, fire once at the statement level. That is, a single INSERT statement might insert 500 rows into a table, but an insert trigger on that table fires only one time. Some vendors allow a trigger to fire for each row of the data-modification operation. So, a statement that inserts 500 rows into a table that has a row-level insert trigger fires 500

In addition to being associated with a specific data-modification statement (INSERT, UPDATE, or DELETE) on a given table, triggers are associated with a specific time of firing. In general, triggers can fire BEFORE the data-modification statement is processed, AFTER it is processed, or (when supported by the vendor) INSTEAD OF processing the statement.

Triggers that fire before or instead of the data-modification statement do not see the changes that the statement renders, while those that fire afterwards can see and act upon the changes that the data-modification statement renders. It may be noted that Oracle allows INSTEAD OF triggers to process only against views, not /tables.

Examples:

The first trigger is invoked whenever there is an update operation on the emp table rows and if salary is changed. It records these changes in another table to know the changes made to the table. The second trigger is invoked whenever there is an insert, delete or update operation on the emp1 table rows and the changes are recorded in the empl and empl1 tables.

```
CREATE OR REPLACE TRIGGER EMPT1
 BEFORE UPDATE ON EMP
 FOR EACH ROW
 WHEN (NEW.SAL <> OLD.SAL)
 BEGIN
 INSERT INTO EMPLOG VALUES ('UPDATED', :OLD.EMPNO,
:OLD.SAL,   :NEW.EMPNO,:NEW.SAL);
 END;
```

```
 CREATE OR REPLACE  TRIGGER EMPT2
 AFTER DELETE OR UPDATE OR INSERT ON EMP1
 FOR EACH ROW
 BEGIN
 IF DELETING THEN
 INSERT INTO EMPL VALUES ('DELETED', :OLD.EMPNO, :OLD.SAL);
 ELSIF INSERTING THEN
 INSERT INTO EMPL VALUES ('ISERTED',:NEW.EMPNO,:NEW.SAL);
 ELSE
 INSERT INTO EMPL1 VALUES
('UPDATED',:OLD.EMPNO,:OLD.SAL,:NEW.EMPNO,:NEW.SAL);
 END IF;
  END;
```

Views:

A view is a table whose rows are not explicitly stored in the database, but are computed as needed from the view definition.

The syntax of creating a view is

```
CREATE [OR REPLACE]  VIEW   view_name
```

```
[(column [,...n])]
AS
SELECT_statement
[WITH CHECK OPTION]
```

Example:

This command creates a view for dbms teacher to view student name and marks in dbms. But the original table contains sno, sname and all subject marks.

```
CREATE VIEW dbms (name, dbmarks)
AS SELECT sname,s1
FROM Students;
```

The syntax of deleting view is

```
DROP VIEW view_name;
```

.

In SQL view is said to be updatable (i.e. insert, update or delete) if the following conditions are all satisfied.

- ✓ The from clause has only one relation.
- ✓ The select clause contains only attributes of the relation and does not have any expressions, aggregate functions, or distinct specification.
- ✓ Any attribute not in the select clause can be set to null.

Another point we can note is that if we insert a row which does not satisfy condition in the where clause of create view then it is allowed and that row doesn't appear in the view. To avoid such insertions we have to use with check option in the create view.

Sequence:

The following command creates a sequence.

```
CREATE SEQUENCE sample_sequence
START WITH 0
MAXVALUE 20
INCREMENT BY 5;
```

A sequence has two attributes, NEXTVAL and CURRVAL.

Sequence Attribute	Description
sequence_name.NEXTVAL	This evaluates to the next highest value.
sequence_name.CURRVAL	This evaluates to the value that was returned from the most recent NEXTVAL call.

Questions:-

1) Create a sequence ‘seq1’ which creates the sequence roll numbers of your class students.
2) Using the above sequence insert values in the dbms table which has the fields sno, name, smarks (at least ten records).
3) Create a trigger on ‘dbms’ table whenever there is an insertion of a row. If the marks are <0 insert 0 and insert 100 when marks is >100.
4) Create a trigger which inserts the changes made into a dbms_change table on dbms table before there is a deletion or modification of a row.
5) Create a trigger on account table which is invoked whenever an account is created or an amount is withdrawn such that the balance should be always greater than or equal to 500.
6) Create a trigger on account table which is invoked after an amount is deposited to display the amount of balance in the account with account number.
7) Create a view for a subject teacher (name,s1) from student table which has the fields roll number, name, all subject marks i.e.s1,s2.. students. Try to insert a row in to the view.
8) Create a view for head of dept table which contains roll number and all subjects’ marks of students from student table. Insert a row in to the view.

Exp no:6 **Date:**

Creation of Database in MS-Access

Aim:- To familiarize with MS Access, open an existing database and create new data base .

Theory:-

MS Access is a relational database that facilitates the storage and retrieval of structured information. Access databases are, ideally, a set of tables that are related in one way or another. Along with tables, Access allows us to create:

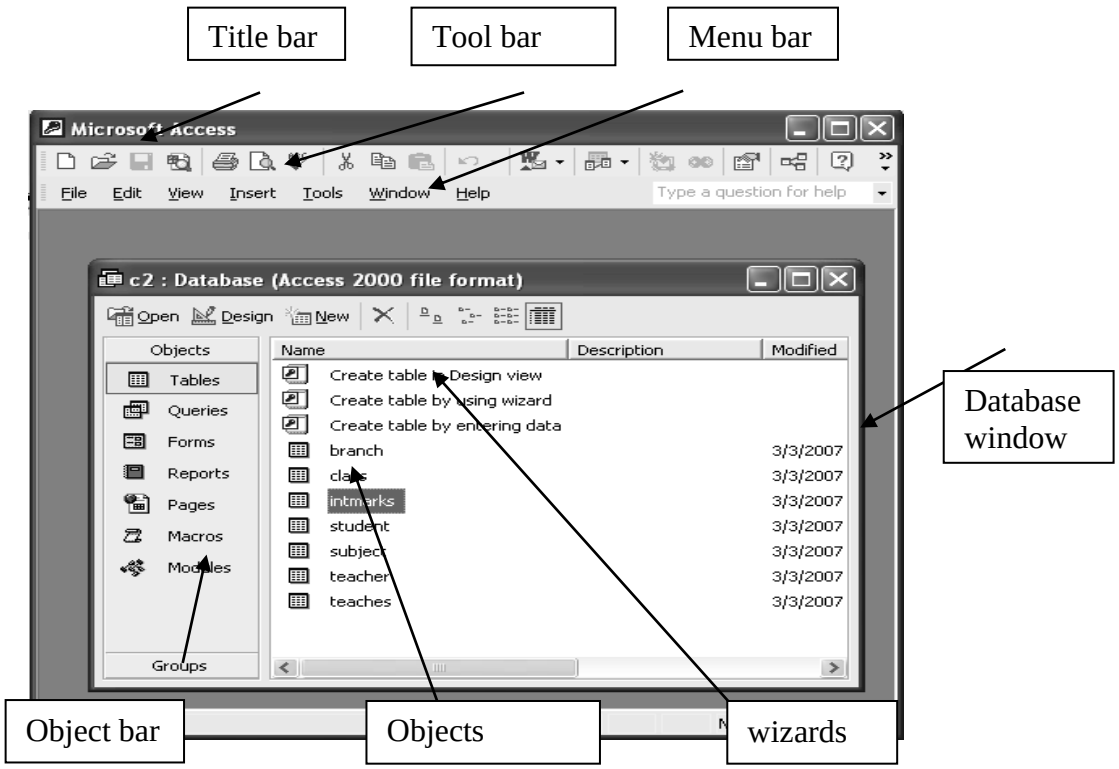
- •Query - organizing or collecting a particular set of data
- •Form - interfaces that allow you to maintain or edit records/data
- •Report - printable results
- •Macros - extended functionality for the database, an advanced function

All of these objects are stored in a single file with ".mdb" file extension.

Procedure:-

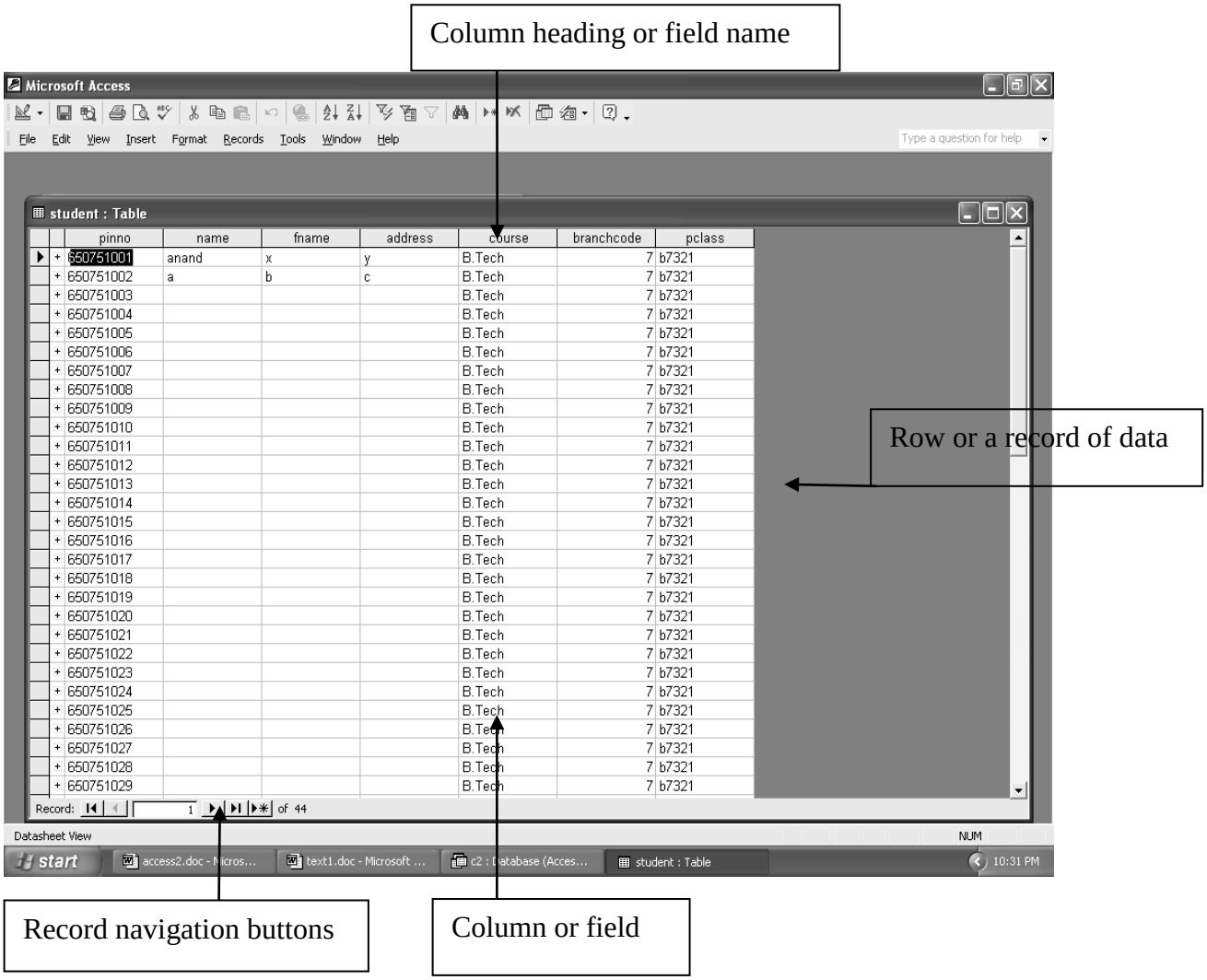
PartI Opening an Existing database:

1) Click on the Windows Start button, select All Programs, and then click Microsoft Access.
2) Click on the FILE menu, then open and select the .mdb file to be opened and then click open.Then the following screen appears.



3) Double click the table to open the table.
4) The table contents are displayed as shown below.
5) Then we navigate to the required record using the record navigation buttons.
6) To add a record of data move to the first empty record and type in the data values and press TAB to move to the next field.
7) To save a record of data move to the first empty field below these records. We do not require to do anything else to save your data. When we leave a record, either by moving to another record or by closing a table, Microsoft access automatically saves the data.
8) Edit a record after moving to the required record and use **TAB and SHIFT + TAB** to move to next field and previous field.
9) Go to **FILE** menu and click exit to exit from Access.

Data Sheet View

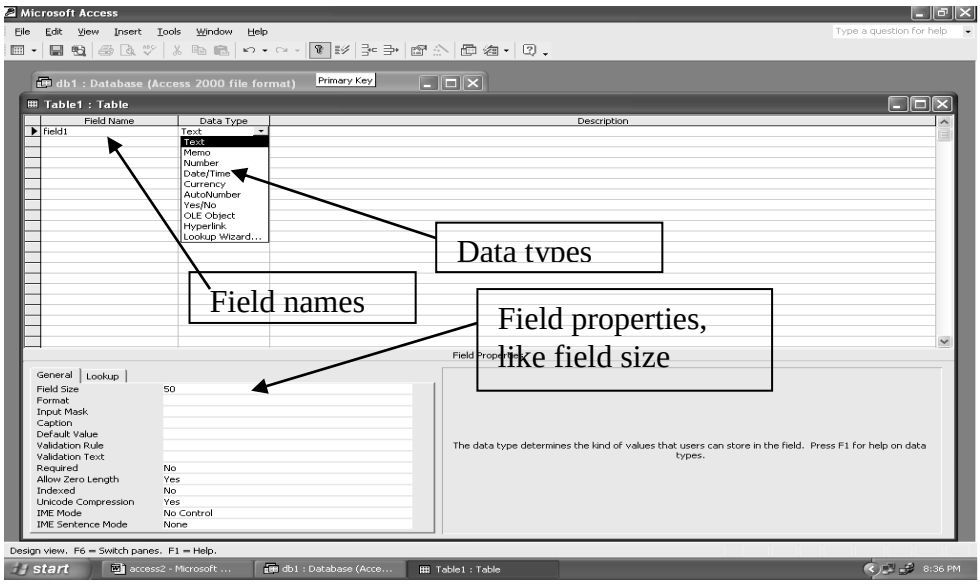


PartII Creating a new database:

1) Click on the Windows **Start** button, select **All Programs**, and then click **Microsoft Access.**
2) Click on the **FILE** menu, then **new**.
3) Click blank database option.
4) Name and save the database.
5) Create table by double-clicking the **create table in design view** option**.** Then the following window appears.
6) Enter the field names and data types.
7) Enter the field width in the field property window.
8) Right click the field which uniquely identifies a record in the table and click primary key option.
9) Set the other properties like
   a) Required yes or no depending on whether the field value can be set to null or not null.
   b) Validation rule like <=100 for ensuring value in the marks field to be not greater than 100.
10) Now save it with a new table name for example table1.
11) Close the design view and double click the table1 object to enter the data.
12) After editing is completed close the data sheet view and exit from MS Access.

Note:-we can also create table using create table by wizard or create table by entering data.

Design view of table



Design view of the table

Data Type Property of a table field

You can use the Data Type property to specify the type of data stored in a table field. Each field can store data consisting of only a single data type.

The **Data Type** property uses the following settings.

Setting	Type of data	Size
Text	(Default) Text or combinations of text and numbers, as well as numbers that don't require calculations, such as phone numbers.	Up to 255 characters or the length set by the Field Size property, whichever is less. Microsoft Access does not reserve space for unused portions of a text field.
Memo	Lengthy text or combinations of text and numbers.	Up to 65,535 characters. (If the Memo field is manipulated through DAO and only text and numbers [not binary data] will be stored in it, then the size of the Memo field is limited by the size of the database.)
Number	Numeric data used in mathematical calculations	1, 2, 4, or 8 bytes (16 bytes if the Field Size property is set to Replication ID).
Date/Time	Date and time values for the years 100 through 9999.	8 bytes.
Currency	Currency values and numeric data used in mathematical calculations involving data with one to four decimal places. Accurate to 15 digits on the left side of the	8 bytes.

	decimal separator and to 4 digits on the right side.	
AutoNumber	A unique sequential (incremented by 1) number or random number assigned by Microsoft Access whenever a new record is added to a table. AutoNumber fields can't be updated. For more information, see the New Values property topic.	4 bytes (16 bytes if the FieldSize property is set to Replication ID).
Yes/No	Yes and No values and fields that contain only one of two values (Yes/No, True/False, or On/Off).	1 bit.
OLE Object	An object (such as a Microsoft Excel spreadsheet, a Microsoft Word document, graphics, sounds, or other binary data) linked to or embedded in a Microsoft Access table.	Up to 1 gigabyte (limited by available disk space)
Hyperlink	Text or combinations of text and numbers stored as text and used as a hyperlink address. A hyperlink address can have up to three parts: text to display — the text that appears in a field or control. address — the path to a file (UNC path) or page (URL). subaddress — a location within the file or page. screentip — the text displayed as a tooltip. The easiest way to insert a hyperlink address in a field or	Each part of the three parts of a Hyperlink data type can contain up to 2048 characters.

	control is to click **Hyperlink** on the **Insert** menu.	
Lookup Wizard	Creates a field that allows you to choose a value from another table or from a list of values by using a list box or combo box. Clicking this option starts the Lookup Wizard, which creates a Lookup field. After you complete the wizard, Microsoft Access sets the data type based on the values selected in the wizard.	The same size as the primary key field used to perform the lookup, typically 4 bytes.

Questions:-

1. Create a table student with roll number, name, father name as fields with appropriate data types.
2. Create a table marks with roll number and marks in different subjects as fields with appropriate data types.
3. Add address field to student table, increase the size allocated to one of the fields.
4. Enter the appropriate data in the tables.

Exp no:7 **Date:**

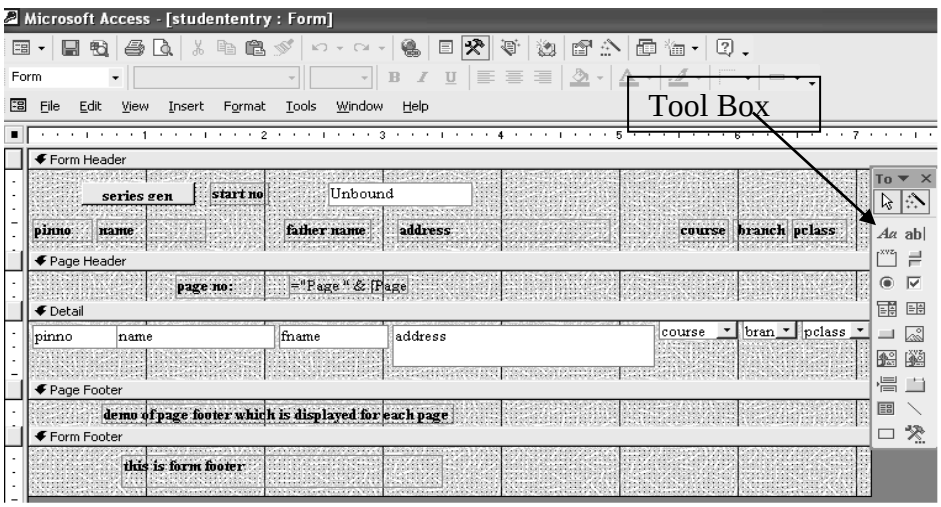
Creation of Forms & Reports in MS-Access

Aim: - To create forms, reports, queries and macros in MS-Access.

Theory:- The form is a formatted display where data is entered, displayed and edited. It provides greater flexibility than a table. A report is a like a query but is formatted for printing. As the data is entered data into a form simultaneously data is adding to the table. In many cases the tables can become quite large and difficult to read, especially when we only want information from individual records. An easier way to view the information in a table is to use a form which is easily created in Access. Microsoft Access divides a form into five sections in the Design view.

- The **form header** prints at the top of the first page .
- When we are viewing data, the form header appears once at the top of the window.
- The **page header** prints at the top of every page .
- The page header only appears when printed or in print preview.
- The **detail** section contains the fields from the table. When we are viewing data, the detail section is repeated for each record. When we print the form, the detail section shows as many records as will fit on a page.
- The **page footer** prints at the bottom of every page .
- The page footer only appears when printed or in print preview.
- The **form footer** prints at the bottom of every page .
- When we are viewing data, the footer appears once at the bottom of the window.

Design view of Form:



We can use a report to present data in print. With a report, we have greater flexibility to present summary information than with a form. For instance, we can include totals across an entire set of records in a report.

Tool Box:-

The tool box is special kind of tool bar. Control is a graphical user interface object, such as a text box, check box, scroll bar, or command button, that lets users control the program. We use controls to display data or choices, perform an action, or make the user interface easier to read Open a form, report, or data access page in Design view. Toolbox is a set of tools that are available in Design view to add controls to a form, report, or data access page. To create a control using the tool box click the tool for the control we want to create and then drag it and drop it.

Text boxes

We use text boxes on a form, report, or data access to display data from a record source. This type of text box is called a bound text box because it's bound to data in a field. Text boxes can also be unbound. For example, we can create an unbound text box to display the results of a calculation or to accept input from a user. Data in an unbound text box isn't stored anywhere.

Labels

We use labels on a form, report, to display descriptive text such as titles, captions, or brief instructions.

List boxes

The list in a list box consists of rows of data. In a form, a list box can have one or more columns, which can appear with or without headings. If a multiple-column list box is bound .Bound control is a control used on a form, report, or data access page to enter or display the contents of a field in the underlying table, query, or SQL statement. The control's Control Source property stores the field name to which the control is bound. Access stores the values from one of the columns. In a data access page, a list box has one column without a heading.

Combo boxes

A combo box is like a text box and a list box combined, so it requires less room. We can type new values in it, as well as select values from a list.

Command buttons

Command buttons provide us with a way of performing action(s) by simply clicking them. When we choose the button, it not only carries out the appropriate action, it also looks as if it's being pushed in and released. We use a command button on a form to start an action or a set of actions. For example, we can create a command button that opens another form. To make a command button do something on a form, we write an event procedure and attach it to the button's On Click property.

Check boxes

We can use a check box on a form, report, or as a stand-alone to display a Yes/No value from an underlying table, query, or SQL statement.

Option buttons

We can use an option button on a form, report, or data access page as a stand-alone control to display a Yes/No value from an underlying record source. We can also use option buttons in an option group to display values to choose from.

The option group is the frame that surrounds the controls inside it. Only one option in an option group can be selected at a time. If an option group is bound to a field, only the group frame itself is bound not the check boxes, toggle buttons, or option buttons inside the frame. Because the Control Source property of the group frame is set to the field that the option group is bound to, we don't set the Control Source property for each in the option group. Instead, we set the Option Value (form or report) or the Value (data access page) property of each check box, toggle button, or option button. In a form or report, set the control property to a number that's meaningful for the field the group frame is bound to. In a data access page, set the control property to either a number or any text that's meaningful for the field the group frame is bound to. When we select an option in an option group, Access sets the value of the field to which the option group is bound to the value of the selected option's Option Value or Value property.

Toggle buttons

We can use a toggle button on a form as a stand-alone to display a Yes/No value from an underlying record source.

Tabbed pages on forms

We can use a tab control to present several pages of information as a single set. This is especially useful when we're working with many controls

that can be sorted into two or more categories. For example, we might use a tab control on an Employees form to separate employment history and personal information. Information about employment history is displayed on this page. Personal information, such as home address and phone number, is displayed on this page.

Procedure:-

PartI Creation of forms:

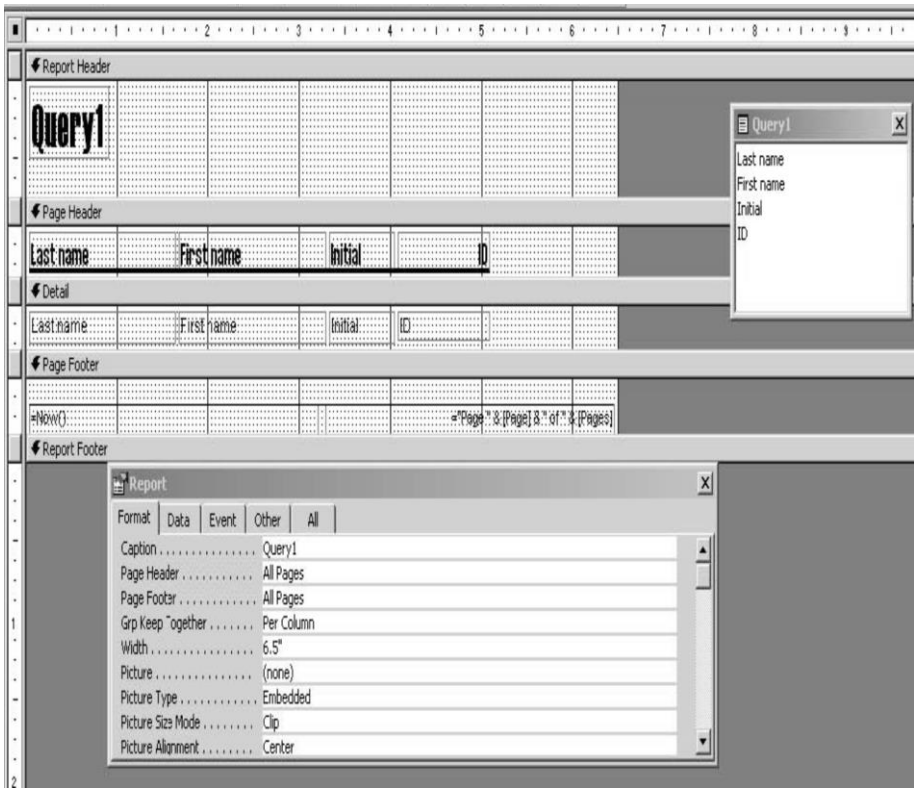
1) Click on **Forms** in the Objects column and select either the wizard or design view
2) The wizard asks which table the form is to come from, and then which fields you want. Give the table name and select the fields and move to the selected field list by clicking right arrow button.
3) Select a layout, style, and a title to complete your form. This then opens in form view for you to begin entering or viewing information.
4) Change to design view to customize.

PartII Creation of reports:

1. Click **Report** in the Object column and select the wizard or design view. The wizard asks which table or query you want to print.
2. Give the table name and then select the fields to print.
3. Select grouping levels **Grouping** brings entries together in the same category, say all people living in Dallas.
4. Select fields to sort and whether to sort ascending or descending order. Up to four fields can be sorted.
5. Then select the layout i.e. columnar, tabular, or justified.
6. Finally, select a style and a title.
7. The report opens in print preview.
8. Switch to design view to change the formatting.

DESIGN VIEW OF A REPORT

When we print the report, sections are repeated, as appropriate, until all the data in the report is printed. The controls in each section tell Microsoft Access what data to print in the section. The Report Header prints at the beginning of the report. The Page Header and Footer prints on every page. The Category Header and Footer prints for each category. The Details section prints for each record in the category Controls that we can use to display, enter, filter, or organize data in Access.



Questions:-

1. Create a form for entering the data into student table which is already created in the expno.7.
2. Create a form for each subject teacher to enter marks in the concerned subject.
3. Create a report to head of the department to view all subject marks of the students.
4. Create a report displaying roll list (roll number and name of the students) from student table.

Exp no:8 **Date:**

Importing Tables

Aim: - To create queries, macros and import a work sheet and a table in oracle to MS Access.

Theory:-Questions including data stored in a Database are called queries. By using queries it is possible to:

- Extract the records that satisfy the search criteria into a record set
- Select which fields are to be displayed
- Perform more complicated searches using logical operators
- Carry out searches on more than one table linked together
- Produce summary statistics
- Calculate new values based on existing fields.

In Microsoft Access, it is easy to automate tasks like changing values in one field when the value in another changes or printing reports by creating macros or writing Visual Basic procedures. A macro automatically carries out a task or a series of tasks for us. Each task that we want Microsoft Access to perform is called an action. Microsoft Access provides a list of actions for us to select from to create a macro. When we run the macro, Microsoft Access carries out the actions in the sequence they are listed, using the objects or data we have specified.

Procedure:-

Part I Creation of query:

1. Open Microsoft Access
2. Click on the Open File button located at the top left of the Access application.
3. Select the file to be opened, then click **Open**
4. Click on the **Queries** object option that is located on the left side of the Access
5. Click on the **Queries** object option that is located on the left side of the Access database window, and then double click the **Create query in Design view** option.
6. Select a table, highlight the table name and then click the “Add” button.
7. Then go to view and click SQL view.

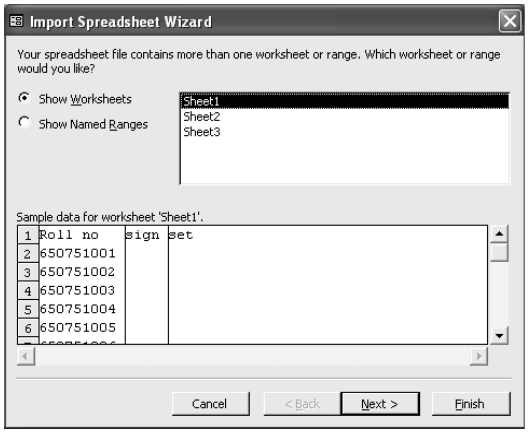
8. Type an SQL query, click save and give query name close select query window.
9. Double click the newly created query to see the result.
10. In stead of going to SQL view we can as well add the fields of a table to SQL design view and can enter the criteria.
11. We attach the query to any push button in the forms to display the result whenever the button is pressed or we can as well create a report for the query

Part II Creation of Macro:

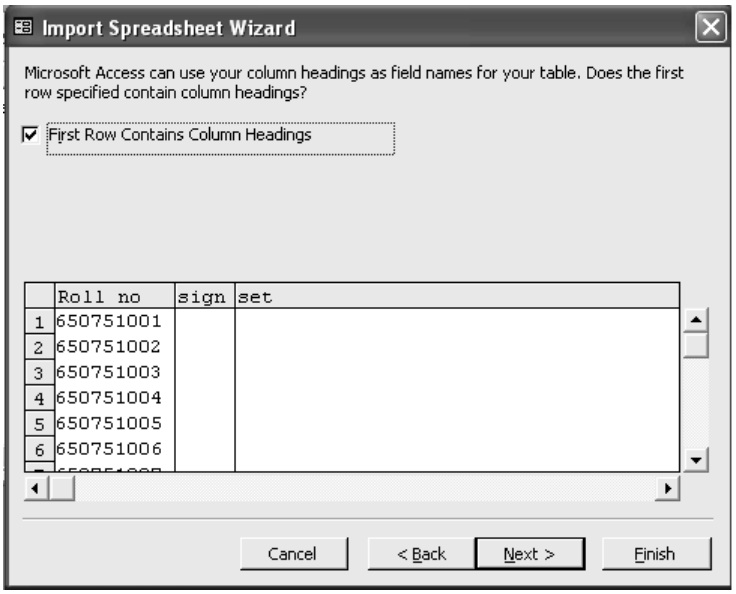
1. Click reports option in the database window and click new.
2. Select an Action for example **Run SQL.**
3. Write **SQL** statement, for example Create table t1(attribute1 number);
4. Click save and give name of the macro.
5. Now double click the newly created macro to execute it so that the new table created.

Part II Import EXEL worksheet:

1. Open Microsoft Excel.
2. In sheet1 type **RollNumber** in the first cell and **Name** in the second cell of first row.
3. Type the roll numbers of students and names of your class mates in the first two columns of consequent rows.
4. Save the work book with the name as **Pinlist**.
5. Close Microsoft Excel.
6. Open Microsoft Access.
7. Open a new database.
8. Click **get external** data in **file** menu and then click **import**.
9. Select the work sheet which is already created after selecting file type as **MS excel** and click import. Then the following window appears.



10. Select work sheet **sheet1** and click next.
11. Then the following window appears.



12. Check the option First row contains columns to make the first column of work sheet as column names of the converted table.
13. And then **select new table** in the next displayed window.
14. Click next and next and choose **my own primary key and** select roll number.
15. Click next and give the table name and click finish.
16. Double click the newly created table to check whether the conversion is completed.

Part IV Import an Oracle table:

1. Open Microsoft Access and open a new database.
2. Click **get external** data in **file** menu and then click **import**.
3. Select file type as **ODBC databases.**
4. Click **Machine database source** tab and then click new.
5. Select **User data source** in the next window and click next.
6. Select **Microsoft odbc drivers for oracle** and Click finish.
7. Give any new name for data source name like ds1 and description as ds1.
8. User name as **scott** and server as oracle9.
9. Click ok and then it asks for password. Give the password as tiger and the click ok.

10. After few minutes all the tables of scott are displayed. Then select one or more tables and click ok to import the tables to Access.

Questions::

1. Create a query to display total marks of each student and execute it
2. Create a query to display average marks in each subject. and execute it
3. Create a macro to run an SQL statement.
4. Import an Excel work sheet into MS access as a table.
5. Import dept and emp tables from oracle.

Exp no: 9 **Date:**

MINI PROJECT

Aim: To develop a database application using DBMS concepts

Procedure:

1. Requirements Analysis:- This step includes obtaining the problem statement from user requirements
2. Conceptual Database Design:- This step includes the following tasks
    a. Identifying entities by noun phrase approach
    b. Indentifying attributes
    c. Identifying relationships
    d. Drawing ER Diagrams
    e. Refining ER diagrams
3. Logical Database Design:- This step includes mapping of ER diagrams to tables
4. Schema Refinement:- This step includes normalization
5. Creation Of Tables:- This step includes creation of tables in Oracle
6. Insertion Of Data:- This step includes insertion of proper data for the project.
7. Prepare SQL quires that are necessary for different users of the mini project.
8. Create triggers, procedures and functions for the mini project.

Exp no: 10 **Date:**

MySQL

Aim: To practice simple MySQL commands for retrieving data.

Description:

MySQL is primarily an RDBMS and ships with no GUI tools to administer MySQL databases or manage data contained within the databases. Users may use the included command line tools,

MySQL uses all the standard ANSI SQL numeric data types, so if you're coming to MySQL from a different database system, these definitions will look familiar to you. The following list shows the common numeric data types and their descriptions.

Features:

Numeric Data Types:

- INT - A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
- TINYINT - A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.
- SMALLINT - A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.
- MEDIUMINT - A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.
- BIGINT - A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 11 digits.
- FLOAT(M,D) - A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.
- DOUBLE(M,D) - A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not

required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.

- DECIMAL(M,D) - An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

Date and Time Types:

- DATE - A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30th, 1973 would be stored as 1973-12-30.
- DATETIME - A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th, 1973 would be stored as 1973-12-30 15:30:00.
- TIMESTAMP - A timestamp between midnight, January 1, 1970 and sometime in 2037. This looks like the previous DATETIME format, only without the hyphens between numbers; 3:30 in the afternoon on December 30th, 1973 would be stored as 19731230153000 (YYYYMMDDHHMMSS).
- TIME - Stores the time in HH:MM:SS format.
- YEAR(M) - Stores a year in 2-digit or 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be 1970 to 2069 (70 to 69). If the length is specified as 4, YEAR can be 1901 to 2155. The default length is 4.

String Types:

Although numeric and date types are fun, most data you'll store will be in string format. This list describes the common string datatypes in MySQL.

- CHAR(M)- A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.
- VARCHAR(M) - A variable-length string between 1 and 255 characters in length; for example VARCHAR(25). You must define a length when creating a VARCHAR field.
- BLOB or TEXT - A field with a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data; the difference between the two is that sorts and comparisons on stored data are case sensitive on BLOBs and are not case sensitive in TEXT fields. You do not specify a length with BLOB or TEXT.

- TINYBLOB or TINYTEXT - A BLOB or TEXT column with a maximum length of 255 characters. You do not specify a length with TINYBLOB or TINYTEXT.
- MEDIUMBLOB or MEDIUMTEXT - A BLOB or TEXT column with a maximum length of 16777215 characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.
- LONGBLOB or LONGTEXT - A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.

ENUM - An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

SYNTAX FOR PROCEDURE AND FUNCTION MySQL

```
CREATE
    [DEFINER = { user | CURRENT_USER }]
    PROCEDURE sp_name ([proc_parameter[,...]])
    [characteristic ...] routine_body

CREATE
    [DEFINER = { user | CURRENT_USER }]
    FUNCTION sp_name ([func_parameter[,...]])
    RETURNS type
    [characteristic ...] routine_body
```

func_parameter:

```
    param_name type
```

type:

```
    Any valid MySQL data type
```

Characteristic:

```
    COMMENT 'string'
  | LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
```

routine_body:

Valid SQL routine statement

PARAMETERS OF PROCEDURE

The real benefit of a stored procedure is of course when you can pass values to it, as well as receive values back. The concept of parameters should be familiar to anyone who has had experience with any procedural programming experience.

There are three types of parameter:

- IN: The default. This parameter is passed to the procedure, and can change inside the procedure, but remains unchanged outside.
- OUT: No value is supplied to the procedure (it is assumed to be NULL), but it can be modified inside the procedure, and is available outside the procedure.
- INOUT: The characteristics of both IN and OUT parameters. A value can be passed to the procedure, modified there as well as passed back again.

Example of database creation in MySQL:

Creating a new user and giving all privileges to that user to access database.

C:\xampp\mysql>cd bin

C:\xampp\mysql\bin>mysql –u root

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 1

Server version:5.1.41 Source distribution

Type ‘help;’ or ‘\h’ fro help. Type ‘\c’ to clear the current input statement

mysql>show databases;

The contents of the databases will be displayed on the screen as follows:

Database

information_schema

cdcol

mysql

phpmyadmin

test

5 rows in set (0.03 sec)

```
Mysql>use mysql;
Database changed
mysql>show tables;
```

Tables in mysql

```
columns_priv
db
event
tables_priv
time_zone
time_zone_leap_second
time_zone_name
time_zone_tranition
time_zone_transition_type
user
23 rows in set(0.05 sec)
```

To create a user in mysql :

```
mysql>insert into user(host,user,passwork,select_priv)
values(‘’,’raju’,password(‘raju’),’Y’);
Query OK, 1 row affected, 3 warnings(0.01 sec)
mysql>select user,password from user;
```

user	**password**
root	
pma	
raju	*B8AEE0A5B33E3547A7D9D016522A18DFCAC6E2AC

```
3 rows in set (0.00 sec)

mysql>flush privileges;
Query OK, 0 rows affected(0.00 sec)

Mysql>create database db;
Query OK, 1 row affected (0.00 sec)

mysql>show databases;
```

Database
information_schema
cdcol

```
mysql
phpmyadmin
test
db
6  rows in set (0.00 sec)

mysql>grant usage on db.* to raju@localhost identified by ‘raju’;
Query OK, 0 rows affected (0.00 sec)

mysql>grant all on sam.* to raju@localhost;
Query OK,0 rows affected (0.00 sec)

C:\xampp\mysql\bin>mysql –u raju –p
Enter password:****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2
Serverversion:5.1.41 Source distribution
Type ‘help;’ or ‘\h’ for help. Type ‘\c’ to clear the current input statement.
mysql>show databases;
Database
information_schema
db
2 rows in set (0.00 sec)
```

Example of CREATION AND USING PROCEDURES AND FUNCTIONS in MySQL:

I. Creating a procedure.

```
mysql>CREATE PROCEDURE sp_in(p VARCHAR(10)) SET @x=P;
Query OK, 0 rows affected (0.00 sec)

mysql>CALL sp_in(‘CSE_BTECH’)
Query OK, 0 rows affected (0.00 sec)

mysql>SELECT @X;
@X
CSE_BTECH
1 row in set (0.00 sec)
```

An OUT using example

```
mysql>CREATE PROCEDURE sp_out(OUT p VARCHAR(10)) SET
p='5th sem';
Query OK, o rows affected (0.00 sec)

mysql>CALL sp_out(@X);
Query OK, 0 rows affected(0.00 sec)

mMysql>select @X;
@X
5th sem
1 row in set (0.00 sec)
(or)
mysql>SET @X='5th  sem';
Query OK, 0 rows affected (0.00 sec)

Mysql>select @X;
@X
5th sem
1 row in set (0.00 sec)
```

II. Creating a Function

```
mysql>CREATE FUNCTION hello (s CHAR(20))
        RETURNS CHAR(50) DETERMINISTIC
        RETURN CONCAT(s);
Query OK, 0 rows affected (0.00 sec)

mysql>SELECT HELLO('CSE");
HELLO('CST')
CST
1 row in set (0.00 sec)
```

Practice Questions:

I. Practice the questions in Experiment No.1 SQL queries in the MYSQL environment.

II. Practice the questions in Experiment No. 2 for creation of tables, modification of table structure and inserting values into the tables in the MYSQL environment also.

III. Practice the questions in Experiment No 3 for practicing SQL set operations, joins and nested queries in the MYSQL environment also.