Department

Of

Computer Engineering

Data Base Management Systems Lab Manual

DILKAP RESEARCH INSTITUTE OF ENGINEERING AND MANAGEMENT STUDIES

Department of Computer Engineering

In-charge **HOD** **Principal**

INDEX

LAB OBJECTIVE

Upon successful completion of this Lab the student will be able to:

- Creating database objects
- Modifying database objects
- Manipulating the data
- Retrieving the data from the database server
- Performing database operations in a procedural manner using pl/sql
- Performing database operations (create, update, modify, retrieve, etc.,) using front-end tools like D2K.
- Design and Develop applications like banking, reservation system, etc.,

EXP 1

AIM: To study and implement DDL and DML commands.

Background Theory

Oracle workgroup or server is the largest selling RDBMS product.it is estimated that the combined sales of both these oracle database product account for aroud 80% of the RDBMSsystems sold worldwide.
These products are constantly undergoing change and evolving. The natural language of this RDBMS product is ANSI SQL,PL/SQL a superset of ANSI SQL.oracle 8i and 9i also under stand SQLJ.

Oracle corp has also incorporated a full-fledged java virtual machine into its database engine.since both executable share the same memory space the JVM can communicate With the database engine with ease and has direct access to oracle tables and their data.

SQL is structure query language.SQL contains different data types those are

1. char(size)
2. varchar2(size)
3. date
4. number(p,s)
5. long
6. raw/long raw

Different types of commands in SQL:

A). **DDL commands: -** To create a database objects
B). **DML commands: -** To manipulate data of a database objects
C). **DQL command: -** To retrieve the data from a database.
D). **DCL/DTL commands: -** To control the data of a database…

DDL commands:

1. The Create Table Command: - it defines each column of the table uniquely. Each column has minimum of three attributes, a name , data type and size.

Syntax:
Create table <table name> (<col1> <datatype>(<size>),<col2> <datatype><size>));

Ex:

create table emp(empno number(4) primary key, ename char(10));

2. Modifying the structure of tables.

a)add new columns

Syntax:

Alter table <tablename> add(<new col><datatype(size),<new col>datatype(size));

Ex:

alter table emp add(sal number(7,2));

3. Dropping a column from a table.

Syntax:

Alter table <tablename> drop column <col>;

Ex:

alter table emp drop column sal;

4. Modifying existing columns.

Syntax:

Alter table <tablename> modify(<col><newdatatype>(<newsize>));

Ex:

alter table emp modify(ename varchar2(15));

5. Renaming the tables

Syntax:

Rename <oldtable> to <new table>;

Ex:

rename emp to emp1;

6. truncating the tables.

Syntax:

Truncate table <tablename>;

Ex:

trunc table emp1;

7. Destroying tables.

Syntax:
Drop table <tablename>;

Ex:
drop table emp;

DML commands:

8. Inserting Data into Tables: - once a table is created the most natural thing to do is load this table with data to be manipulated later.

Syntax:
insert into <tablename> (<col1>,<col2>) values(<exp>,<exp>);

9. Delete operations.

a) remove all rows
Syntax:
delete from <tablename>;

b) removal of a specified row/s
Syntax:
delete from <tablename> where <condition>;

10. Updating the contents of a table.

a) updating all rows
Syntax:
Update <tablename> set <col>=<exp>,<col>=<exp>;

b) updating seleted records.
Syntax:
Update <tablename> set <col>=<exp>,<col>=<exp> where <condition>;

11. Types of data constrains.
a) not null constraint at column level.
Syntax:
<col><datatype>(size)not null

b) unique constraint
Syntax:
Unique constraint at column level.
<col><datatype>(size)unique;

c) unique constraint at table level:

Syntax:
Create table tablename(col=format,col=format,unique(<col1>,<col2>);

d) primary key constraint at column level
Syntax:
<col><datatype>(size)primary key;

e) primary key constraint at table level.
Syntax:
Create table tablename(col=format,col=format
primary key(col1>,<col2>);

f) foreign key constraint at column level.
Syntax:
<col><datatype>(size>) references <tablename>[<col>];

g) foreign key constraint at table level
Syntax:
foreign key(<col>[,<col>])references <tablename>[(<col>,<col>)

h) check constraint
check constraint constraint at column level.
Syntax: <col><datatype>(size) check(<logical expression>)

i) check constraint constraint at table level.
Syntax: check(<logical expression>)

DQL Commands:

12. Viewing data in the tables: - once data has been inserted into a table, the next most logical operation would be to view what has been inserted.

a) all rows and all columns
Syntax:

Select <col> to <col n> from tablename;

Select * from tablename;

13. Filtering table data: - while viewing data from a table, it is rare that all the data from table will be required each time. Hence, sql must give us a method of filtering out data that is not required data.

a) Selected columns and all rows:
Syntax:
select <col1>,<col2> from <tablename>;

b) selected rows and all columns:
Syntax:
select * from <tablename> where <condition>;

c) selected columns and selected rows
Syntax:
select <col1>,<col2> from <tablename> where<condition>;

14. Sorting data in a table.
Syntax:
Select * from <tablename> order by <col1>,<col2> <[sortorder]>;

DCL commands:

Oracle provides extensive feature in order to safeguard information stored in its tables from unauthoraised viewing and damage.The rights that allow the user of some or all oracle resources on the server are called privileges.

a) Grant privileges using the GRANT statement

The grant statement provides various types of access to database objects such as tables,views and sequences and so on.

Syntax:
GRANT <object privileges>
ON <objectname>
TO<username>
[WITH GRANT OPTION];

b) Reoke permissions using the REVOKE statement:

The REVOKE statement is used to deny the Grant given on an object.

Syntax:
REVOKE<object privilege>
ON
FROM<user name>;

EXP 2

Aim: To implement integrity constraints.

Theory:

What is SQL?
Ans: Structured Query Language

What is database?

A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

What is DBMS?

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of *defining*, *constructing* and *manipulating* the database for various applications.

What is a Database system?

The database and DBMS software together is called as Database system.

Advantages of DBMS?

- Redundancy is controlled.
- Unauthorised access is restricted.
- Providing multiple user interfaces.
- Enforcing integrity constraints.
- Providing backup and recovery.

Disadvantage in File Processing System?

- Data redundancy & inconsistency.
- Difficult in accessing data.
- Data isolation.
- Data integrity.
- Concurrent access is not possible.
- Security Problems.

Create the tables with the appropriate integrity constraints

Insert around 10 records in each of the tables

```
SQL>create table student(stud_no number(5) primary key,stud_name
varchar2(15));

SQL>desc student;
  Name                              Null?    Type
……..
STUD_NO                                NOT NULL  NUMBER(5)
STUD_NAME                                        VARCAHR2(15)
```

Valid Test Data:

```
SQL>insert into student values(&stud_no,'&stud_name');

SQL>select * from student;

STUD_NO    STUD_NAME
.....

508              HARISH
513              BALAJI
518              RAKESH
524              PAVAN
534              JOYCE

SQL>create table membership(mem_no number(5) primary key,stud_no
number(5) references student(stud)no));
SQL>dsec membership;

Name                              Null?            Type
…….
MEM_NO                         NOT NULL         NUMBER(5)
STUD_NO                                         NUMBER(5)


SQL>insert into membership values(&mem_no,&stud_no);
Enter value for mem_no:5440
Enter value for stud_no:510
old  1:insert into membership values(&mem_no,&stud_no)
new 1:insert into membership values(5440,510)
insert into membership values(5440,510)
*
```

Errors Observed:

```
ERROR at line 1:
ORA-02291:integrity constraint(HARISH.SYS_C002724)violated-primary key not
found

SQL>select * from membership;
```

```
      MEM_NO          STUD_NO
……..
         5440                513
         5441                508
         5442                518
         5443                534
         5444                524
```

SQL>create table book(book_no number(5) primary key,book_name varchar2(20),author varchar2(2));

SQL>desc book;

```
Name                              Null?    Type
……..
BOOK_NO                           NOT NULL         NUMBER(5)
BOOK_NAME                                          VARCHAR2(20)
AUTHOR                                             VARCHAR2(20)
```

SQL>insert into book values(&book_no,'&book_name','&author');
SQL>select * from book;

```
BOOK_NO        BOOK_NAME         AUTHOR
……..
9123             DBMS              Rama Krishna
2342             JAVA              Robett wilkins
4523             Fearless tales    Alfred
8723             my ambition       Harish
7821             Harry Potter      JK Rowling
```

SQL>create table lss_rec(iss_no number primary key,iss_date date,mem_no number(5) references membership(mem_no),book_no number(5) references book(book_no));

SQL>desc iss_rec;

```
Name                              Null?             Type
……
ISS_NO                          NOT NULL          NUMBER
ISS_DATE                                          DATE
MEM_NO                                            NUMBER(5)
BOOK_NO                                           NUMBER(5)
```

SQL>select * from iss_rec;

```
ISS_NO      ISS_DATE      MEM_NO      BOOK_NO
……
43          05-JAN-06       5443              4523
81          28-DEC-05       5441              8723
22          08-DEC-05       5440              7821
```

```
53          07-JAN-06       5442            9123
35          06-JAN-06       5444            2342
```

c) List all the student names with their membership numbers

```
SQL> select s.studname, m.memno from student s, membership m where
m.studno=s.studno;


STUDNAME MEMNO
--------------------
abhijeet            1001
arun                1002
arvind              1003
ashish              1004
ashwin              1005
```

d) List all the issues for the current date with student and Book names

```
SQL> select i.issno, s.studname, b.bookname from iss_rec I, membership m,
student s, book b
2 where i.memno=m.memno and m.studno=s.studno and
i.issdate=to_char(sysdate);

ISSNO       STUDNAME BOOKNAME
-------     ------------        ---------------
13          arvind              P&S
```

e) List the details of students who borrowed book whose author is CJDATE

```
SQL> select * from student where studno in(select studno from membership
where memno in
2 (select memno from iss_rec where bookno in(select bookno from book where
author='CJDATE')));

STUDNO    STUDNAME
----------    -------------
   505          ashwin
```

f) Give a count of how many books have been bought by each student

```
SQL> select s.studno, count(i.bookno) from student s.membership m, book b, 2
iss_rec I where s.studno=m.studno and b.bookno=i.bookno group by s.studno;

STUDNO    COUNT(I.BOOKNO)
----------    ----------------------
501         5
```

```
502         5
503         5
504         5
505         5
```

g) Give a list of books taken by student with stud_no as 5

```
SQL> select bookname from book where bookno in (select bookno from iss_rec
where
2 memno in(select memno from membership where
3 studno in(select studno from student where studno=5)));

BOOKNAME
------------
NT
```

h) List the book details which are issued as of today

```
SQL> delete from book where bookno in(select bookno from iss_rec where
issdate=to_char(sysdate));
delete from book where bookno in (select bookno from iss_rec where
issdate=to_char(sysdate))
```

Errors Observed:

```
ERROR at line 1:
ORA-02292: integrity constraint (SCOTT.SYS_C00840) violated – child record
found
```

i) Create a view which lists out the iss_no, iss _date, stud_name, book name

j) Create a view which lists the daily issues-date wise for the last one week

EXP 3

AIM: To implement pattern matching and set operations.

THEORY:

What is Data Model?

A collection of conceptual tools for describing data, data relationships data semantics and constraints.

What is E-R model?

This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

What is Object Oriented model?

This model is based on collection of objects. An object contains values stored in instance variables with in the object. An object also contains bodies of code that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

What is an Entity?

It is a 'thing' in the real world with an independent existence.

What is an Entity type?

It is a collection (set) of entities that have same attributes.

What is an Entity set?

It is a collection of all entities of particular entity type in the database.

SQL> create table Depositor(Customer_name varchar2(20), Account_no number(5));

Table created.

SQL> create table Borrower2(Customer_name varchar2(20), Loan_no number(5));

Table created.

SQL> insert into Depositor values('john',1001);

1 row created.

```
SQL> insert into Depositor values('sita',1002);

1 row created.

SQL> insert into Depositor values('vishal',1003);

1 row created.

SQL> insert into Depositor values('ram',1004);

1 row created.

SQL> insert into Borrower2 values('john',2001);

1 row created.

SQL> insert into Borrower2 values('tonny',2003);

1 row created.

SQL> insert into Borrower2 values('rohit',2004);

1 row created.

SQL> insert into Borrower2 values('vishal',2005);

1 row created.

SQL> select * from Depositor;

CUSTOMER_NAME       ACCOUNT_NO
-------------------- ----------
john                  1001
sita                 1002
vishal                1003
ram                   1004

SQL> select * from Borrower2;

CUSTOMER_NAME         LOAN_NO
-------------------- ----------
john                  2001
tonny                  2003
rohit                 2004
vishal                2005

SQL> select Customer_name
```

```
  2  from Borrower2
  3  union
  4  select Customer_name
  5  from Depositor;

CUSTOMER_NAME
--------------------
john
ram
rohit
sita
tonny
vishal

6 rows selected.

SQL> select Customer_name
  2  from Borrower2
  3  union all
  4  select Customer_name
  5  from Depositor;

CUSTOMER_NAME
--------------------
john
tonny
rohit
vishal
john
sita
vishal
ram

8 rows selected.

SQL> select Customer_name
  2  from Depositor
  3  intersect
  4  select Customer_name
  5  from Borrower2;

CUSTOMER_NAME
--------------------
john
vishal

SQL> select Customer_name
```

```
  2  from Depositor
  3  intersect all
  4  select Customer_name
  5  from Borrower2;

CUSTOMER_NAME
--------------------
john
vishal
john
vishal

SQL> select Customer_name
  2  from Depositor
  3  minus
  4  select Customer_name
  5  from Borrower2;

CUSTOMER_NAME
--------------------
ram
sita

SQL> create table author1(Title varchar2(20), Author_name
varchar2(20),
  2  Publisher_year number(5), Publisher_name varchar2(20));

Table created.

SQL> insert into author1 values('Oracle','Arora',2004,'PHI');

1 row created.

SQL> insert into author1 values('DBMS','Basu',2004,'Technical');

1 row created.

SQL> insert into author1 values('DOS','Sinha',2003,'Nirali');

1 row created.

SQL> insert into author1 values('ADBMS','Basu',2004,'Technical');

1 row created.

SQL> insert into author1 values('UNIX','Kapoor',2000,'Sci-Tech');
```

1 row created.

```
TITLE                          AUTHOR_NAME                    PUBLISHER_YEAR
PUBLISHER_NAME
-------------------- -------------------- -------------- --------------------
Oracle               Arora                          2004 PHI
DBMS                 Basu                           2004 Technical
DOS                  Sinha                          2003 Nirali
ADBMS                Basu                           2004 Technical
UNIX                 Kapoor                         2000 Sci-Tech
```

```
SQL> select Author_name
  2  from author1
  3  where Author_name like 'Ba%';

AUTHOR_NAME
--------------------
Basu
Basu
```

```
SQL> select Author_name
  2  from author1
  3  where Author_name like '_r%' or Author_name like '_a%';

AUTHOR_NAME
--------------------
Arora
Basu
Basu
Kapoor
```

EXP 4

AIM: To implement aggregate functions.

THEORY:

Aggregate Functions

MIN	returns the smallest value in a given column
MAX	returns the largest value in a given column
SUM	returns the sum of the numeric values in a given column
AVG	returns the average value of a given column
COUNT	returns the total number of values in a given column

COUNT(*)	returns the number of rows in a table

Aggregate functions are used to compute against a "returned column of numeric data" from your SELECT statement. They basically summarize the results of a particular column of selected data. We are covering these here since they are required by the next topic, "GROUP BY". Although they are required for the "GROUP BY" clause, these functions can be used without the "GROUP BY" clause.

```
SQL> create table Book1(Title varchar2(7), ISBN number(5),
  2  Pub_year number(5),Author_name varchar2(10),
  3  Unit_price number(5),Publisher_name varchar2(10));

Table created.

SQL> insert into Book1 values('Oracle',1001,2004,'Arora',399,'PHI');

1 row created.

SQL> insert into Book1
values('DBMS',1002,2004,'Basu',400,'Technical');

1 row created.

SQL> insert into Book1 values('DOS',2001,2003,'Sinha',250,'Nirali');

1 row created.

SQL> insert into Book1
values('ADBMS',2002,2004,'Basu',450,'Technical');

1 row created.

SQL> insert into Book1 values('UNIX',2003,2000,'Kapoor',300,'Sci-
Tech');

1 row created.

SQL> select * from Book1;

TITLE       ISBN   PUB_YEAR AUTHOR_NAM UNIT_PRICE PUBLISHER_
------- ---------- ---------- ---------- ---------- ----------
Oracle       1001       2004 Arora             399 PHI
DBMS         1002       2004 Basu              400 Technical
DOS          2001       2003 Sinha             250 Nirali
ADBMS        2002       2004 Basu              450 Technical
```

```
UNIX        2003      2000 Kapoor          300 Sci-Tech

SQL> select avg(Unit_price)"Average Price"
  2  from Book1;

Average Price
-------------
      359.8

SQL> select min(Unit_price)"Minimum Price"
  2  from Book1;

Minimum Price
-------------
        250

SQL> select max(Unit_price)"Maximum Price"
  2  from Book1;

Maximum Price
-------------
        450

SQL> select sum(Unit_price)"Total"
  2  from Book1;

    Total
----------
     1799

SQL> select count(Title)"No. of Books"
  2  from Book1;

No. of Books
------------
         5
```

EXP 5

AIM: To implement 'where', 'from', 'order by', 'group by', 'having' clause in SQL

Theory

FROM clause

The FROM clause is a mandatory clause in a *SelectExpression*. It specifies the tables (*TableExpression*) from which the other clauses of the query can access columns for use in expressions.

GROUP BY clause

A GROUP BY clause, part of a *SelectExpression*, groups a result into subsets that have matching values for one or more columns. In each group, no two rows have the same value for the grouping column or columns. NULLs are considered equivalent for grouping purposes.

You typically use a GROUP BY clause in conjunction with an aggregate expression.

HAVING clause

A HAVING clause restricts the results of a GROUP BY in a *SelectExpression*. The HAVING clause is applied to each group of the grouped table, much as a WHERE clause is applied to a select list. If there is no GROUP BY clause, the HAVING clause is applied to the entire result as a single group. The SELECT clause cannot refer directly to any column that does not have a GROUP BY clause. It can, however, refer to constants, aggregates, and special registers.

ORDER BY clause

The ORDER BY clause is an optional element of the following:

- A SELECT statement
- A *SelectExpression*
- A VALUES expression
- A *ScalarSubquery*
- A *TableSubquery*

It can also be used in an INSERT statement or a CREATE VIEW statement.

An ORDER BY clause allows you to specify the order in which rows appear in the result set. In subqueries, the ORDER BY clause is meaningless unless it is accompanied by one or both of the result offset and fetch first clauses or in conjunction with the ROW_NUMBER function, since there is no guarantee that the order is retained in the outer result set. It is permissible to combine ORDER BY on the outer query with ORDER BY in subqueries.

WHERE clause

A WHERE clause is an optional part of a *SelectExpression*, DELETE statement, or UPDATE statement. The WHERE clause lets you select rows based on a boolean expression. Only rows for which the expression evaluates to TRUE are returned in the result, or, in the case of a DELETE statement, deleted, or, in the case of an UPDATE statement, updated.

```
SQL> create table Book1(Title varchar2(7), ISBN number(5),
  2  Pub_year number(5),Author_name varchar2(10),
  3    Unit_price number(5),Publisher_name varchar2(10));

Table created.

SQL> insert into Book1
values('Oracle',1001,2004,'Arora',399,'PHI');

1 row created.

SQL> insert into Book1
values('DBMS',1002,2004,'Basu',400,'Technical');

1 row created.

SQL> insert into Book1
values('DOS',2001,2003,'Sinha',250,'Nirali');

1 row created.

SQL> insert into Book1
values('ADBMS',2002,2004,'Basu',450,'Technical');

1 row created.
```

```
SQL> insert into Book1
values('UNIX',2003,2000,'Kapoor',300,'Sci-Tech');

1 row created.

SQL> select * from Book1;

TITLE       ISBN   PUB_YEAR AUTHOR_NAM UNIT_PRICE
PUBLISHER_
------- ---------- ---------- ---------- ---------- ----------
Oracle       1001       2004 Arora             399 PHI
DBMS          1002       2004 Basu              400 Technical
DOS          2001       2003 Sinha             250 Nirali
ADBMS         2002       2004 Basu              450 Technical
UNIX         2003       2000 Kapoor            300 Sci-Tech

SQL> select Title from Book1 where Pub_year='2004';

TITLE
-------
Oracle
DBMS
ADBMS

SQL> select Title from Book1
  2  where Unit_price between 300 and 400;

TITLE
-------
Oracle
DBMS
UNIX

SQL> select Title from Book1
  2  where Unit_price>=300 and Unit_Price<=400;

TITLE
-------
Oracle
DBMS
```

UNIX

```
SQL> select Title,Author_name,Publisher_name
  2  from Book1
  3  where Pub_year='2004';

TITLE   AUTHOR_NAM PUBLISHER_
------- ---------- ----------
Oracle  Arora      PHI
DBMS    Basu       Technical
ADBMS   Basu       Technical

SQL> select Title, Unit_price
  2  from Book1
  3  order by Title;

TITLE   UNIT_PRICE
------- ----------
ADBMS         450
DBMS          400
DOS           250
Oracle        399
UNIX          300

SQL> select Title, Unit_price, Pub_year
  2  from Book1
  3  order by Pub_year desc;

TITLE   UNIT_PRICE   PUB_YEAR
------- ---------- ----------
Oracle        399       2004
DBMS          400       2004
ADBMS         450       2004
DOS           250       2003
UNIX          300       2000

SQL> select Publisher_name, sum(Unit_price)
  2  "Total Book Amount"
  3  from Book1
  4  group by Publisher_name;
```

```
PUBLISHER_ Total Book Amount
---------- -----------------
PHI                399
Technical            850
Nirali             250
Sci-Tech             300

SQL> select Author_name, sum(Unit_price*0.15)"Royalty
Amount"
  2  from Book1
  3  group by Author_name;

AUTHOR_NAM Royalty Amount
---------- --------------
Basu              127.5
Sinha              37.5
Kapoor               45
Arora             59.85

SQL> select Publisher_name, sum(Unit_price)"Total Book
Amount"
  2  from Book1
  3  group by Publisher_name
  4  having Publisher_name<>'PHI';

PUBLISHER_ Total Book Amount
---------- -----------------
Technical            850
Nirali             250
Sci-Tech             300

SQL> select Author_name, sum(Unit_price*0.15)"Royalty
Amount"
  2  from Book1
  3  group by Author_name
  4  having Author_name like '_a%';

AUTHOR_NAM Royalty Amount
---------- --------------
```

Basu 127.5
Kapoor 45

EXP 6

AIM: To implement nested and complex queries

Theory

What is an Extension of entity type?

The collections of entities of a particular entity type are grouped together into an entity set.

What is Weak Entity set?

An entity set may not have sufficient attributes to form a primary key, and its primary key compromises of its partial key and primary key of its parent entity, then it is said to be Weak Entity set.

What is an attribute?

It is a particular property, which describes the entity.

What is a Relation Schema and a Relation?

A relation Schema denoted by R(A1, A2, …, An) is made up of the relation name R and the list of attributes Ai that it contains. A relation is defined as a set of tuples. Let r be the relation which contains set tuples (t1, t2, t3, ..., tn). Each tuple is an ordered list of n-values t=(v1,v2, ..., vn).

What is degree of a Relation?

It is the number of attribute of its relation schema.

What is Relationship?

It is an association among two or more entities.

16. What is Relationship set?

The collection (or set) of similar relationships.

SQL> create table Book1(Title varchar2(7), ISBN number(5),Pub_year number(5),Aut hor_name varchar2(10),Unit_price number(5),Publisher_name varchar2(10));

Table created.

```
SQL> insert into Book1
values('Oracle',1001,2004,'Arora',399,'PHI');

1 row created.

SQL> insert into Book1
values('DBMS',1002,2004,'Basu',400,'Technical');

1 row created.

SQL> insert into Book1
values('DOS',2001,2003,'Sinha',250,'Nirali');

1 row created.

SQL> insert into Book1
values('ADBMS',2002,2004,'Basu',450,'Technical');

1 row created.

SQL> insert into Book1
values('UNIX',2003,2000,'Kapoor',300,'Sci-Tech');

1 row created.

SQL> commit;

Commit complete.

SQL> select*from book1;

TITLE       ISBN   PUB_YEAR AUTHOR_NAM UNIT_PRICE
PUBLISHER_
------- ---------- ---------- ---------- ---------- ----------
Oracle       1001       2004 Arora             399 PHI
DBMS         1002       2004 Basu              400 Technical
DOS          2001       2003 Sinha             250 Nirali
ADBMS        2002       2004 Basu              450 Technical
UNIX         2003       2000 Kapoor            300 Sci-Tech
```

SQL> select Title,Author_name,Publisher_name,Pub_year from BOOK1 where Pub_year in('2000','2002','2004');

TITLE AUTHOR_NAM PUBLISHER_ PUB_YEAR
------- ---------- ---------- ----------
Oracle Arora PHI 2004
DBMS Basu Technical 2004
ADBMS Basu Technical 2004
UNIX Kapoor Sci-Tech 2000

SQL> create table Author1(Author_name varchar2(10),country varchar2(10));

Table created.

SQL> insert into Author1 values('Arora','U.S');

1 row created.

SQL> insert into Author1 values('Kapoor','Canada');

1 row created.

SQL> insert into Author1 values('Basu','India');

1 row created.

SQL> insert into Author1 values('Sinha','India');

1 row created.

SQL> commit;

Commit complete.

SQL> select *from Author1;

AUTHOR_NAM COUNTRY
---------- ----------

```
Arora      U.S
Kapoor     Canada
Basu       India
Sinha      India

SQL> select*from Author1 where Author_name in(select
Author_name from BOOK1 wher
e Pub_year='2004');

AUTHOR_NAM COUNTRY
---------- ----------
Arora      U.S
Basu       India

SQL> select Title,Author_name,Publisher_name,Pub_year
from book1 where Pub_year
not in('2002','2004','2005');

TITLE   AUTHOR_NAM PUBLISHER_   PUB_YEAR
------- ---------- ---------- ----------
DOS     Sinha      Nirali          2003
UNIX    Kapoor     Sci-Tech        2000

SQL> select title from book1 where Author_name not in(select
Author_name from au
thor1 where country='India');

TITLE
-------
Oracle
UNIX

SQL> select distinct B1.title from book1 b1,book1 b2 where
b1.UNit_price>B2.Unit
_price and B2.Pub_year='2004';

TITLE
-------
ADBMS
DBMS
```

SQL> select distinct Title from book1 where Unit_price>some(select Unit_price fr om book1 where Pub_year='2004');

TITLE

ADBMS
DBMS

SQL> Select Author_name from book1 group by Author_name having sum(Unit_price*0. 15)>=all(select sum(Unit_price*0.15) from book1 group by Author_name);

AUTHOR_NAM

Basu

SQL> create table order1(Order_no varchar2(2),Isbn varchar2(7),Order_date varcha r2(10),Qty varchar2(3),Price varchar2(3));

Table created.

SQL> insert into Order1 values(1,1001,'10-10-2004',100,399);

1 row created.

SQL> insert into Order1 values(2,1002,'11-01-2004',50,400);

1 row created.

SQL> commit;

Commit complete.

SQL> select *from Order1;

```
OR ISBN    ORDER_DATE QTY PRI
-- ------- ---------- --- ---
1  1001    10-10-2004 100 399
2  1002    11-01-2004 50  400

SQL> select title from book1 where exists(select *from Order1
where BOOk1.isbn=O
rder1.isbn);

TITLE
-------
Oracle
DBMS

SQL> select title from book1 where not exists(select *from
order1 where book1.is
bn=order1.isbn);

TITLE
-------
UNIX
DOS
ADBMS

----------COMPLEX QUERIES----------

----------DERIVED RELATIONS----------

SQL> CREATE TABLE Sales1
  2 (
  3  Salesperson VARCHAR2(3),
  4  Area CHAR(1),
  5  Value NUMBER(5)
  6 );

Table created.

SQL> INSERT INTO Sales VALUES ('Bob', 'N', 100);
```

1 row created.

SQL> INSERT INTO Sales VALUES ('Bob', 'N', 125);

1 row created.

SQL> INSERT INTO Sales VALUES ('Sam', 'N', 120);

1 row created.

SQL> INSERT INTO Sales VALUES ('Jim', 'S', 120);

1 row created.

SQL> INSERT INTO Sales VALUES ('Tim', 'S', 130);

1 row created.

SQL> Select * from Sales1;

SALESPERSON	AREA	VALUE
Bob	N	100.00
Bob	N	125.00
Sam	N	120.00
Jim	S	120.00
Tim	S	130.00

```
SQL> SELECT * FROM
  1 (
  2 SELECT SalespersonAS SalesDetail, Value FROM Sales
  3 ) AS DerivedSales
  4 WHERE Value > 100;
```

SALESDETAIL	VALUE
Bob	125.00
Sam	120.00
Jim	120.00
Tim	130.00

----------WITH CLAUSE----------

SQL> select * from Account1;

```
BRANCH_NAM    BALANCE
---------- ----------
COMP           10000
IT             20000
COMP           50000
IT             40000
COMP           45000
```

```
SQL> with Max_balance (value) as
  2  select max(Balance)
  3  from Account1
  4  select Branch_name
  5  from Account1,Max_balance
  6  where Account1.Balance=Max_balance value;
```

```
BRANCH_NAM    MAX_BALANCE
---------- ----------
COMP           50000
```

EXP 7

AIM: To implement views in SQL.

Theory

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

What is VDL (View Definition Language)?

It specifies user views and their mappings to the conceptual schema.

What is SDL (Storage Definition Language)?

This language is to specify the internal schema. This language may specify the mapping between two schemas.

What is Data Storage - Definition Language?

The storage structures and access methods used by database system are specified by a set of definition in a special type of DDL called data storage-definition language.

What is DML (Data Manipulation Language)?

This language that enable user to access or manipulate data as organised by appropriate data model.

- *Procedural DML or Low level:* DML requires a user to specify what data are needed and how to get those data.
- *Non-Procedural DML or High level:* DML requires a user to specify what data are needed without specifying how to get those data.

What is DML Compiler?

It translates DML statements in a query language into low-level instruction that the query evaluation engine can understand.

SQL> SELECT * FROM CUSTOMER;

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

```
SQL> CREATE VIEW CUSTOMERS_VIEW AS
  2  SELECT name, age
  3  FROM  CUSTOMERS;

SQL> SELECT * FROM CUSTOMERS_VIEW;

NAME       AGE
-------     ----
Ramesh      32
Khilan      25
kaushik     23
Chaitali    25
Hardik      27
Komal       22
Muffy       24

SQL> SELECT NAME FROM CUSTOMERS_VIEW WHERE
  2  AGE = 25;

NAME
------
Khilan
```

Chaitali

```
SQL> UPDATE CUSTOMERS_VIEW
  2  SET AGE = 35
  3  WHERE name='Ramesh';

ID  NAME      AGE  ADDRESS     SALARY
--- -------    ---- --------    -------
1   Ramesh     35   Ahmedabad    2000.00
2   Khilan     25   Delhi       1500.00
3   kaushik    23   Kota         2000.00
4   Chaitali   25   Mumbai       6500.00
5   Hardik     27   Bhopal       8500.00
6   Komal      22   MP          4500.00
7   Muffy      24   Indore      10000.00

SQL> DELETE FROM CUSTOMERS_VIEW
  2  WHERE age = 22;

ID  NAME      AGE  ADDRESS     SALARY
--- -------    ---- --------    -------
1   Ramesh     35   Ahmedabad    2000.00
```

2 Khilan 25 Delhi 1500.00

3 kaushik 23 Kota 2000.00

4 Chaitali 25 Mumbai 6500.00

5 Hardik 27 Bhopal 8500.00

7 Muffy 24 Indore 10000.00

EXP 8

AIM: To implement join operations.

Theory:

Join in SQL

SQL Join is used to fetch data from two or more tables, which is joined to appear as single set of data. SQL Join is used for combining column from two or more tables by using values common to both tables. **Join** Keyword is used in SQL queries for joining two or more tables. Minimum required condition for joining table, is **(n-1)** where **n**, is number of tables. A table can also join to itself known as, **Self Join**.

Types of Join

The following are the types of JOIN that we can use in SQL.

- Inner
- Outer
- Left
- Right

SQL> SELECT * FROM CUSTOMERS;

ID NAME AGE ADDRESS SALARY

```
-- -------   --- ----------  --------

1  Ramesh    32  Ahmedabad   2000.00

2  Khilan    25  Delhi       1500.00

3  kaushik   23  Kota        2000.00

4  Chaitali  25  Mumbai      6500.00

5  Hardik    27  Bhopal      8500.00

6  Komal     22  MP          4500.00

7  Muffy     24  Indore      10000.00
```

```
SQL> SELECT * FROM ORDERS;

OID  DATE                CUSTOMER_ID  AMOUNT

---  -----               -----------  -------

102  2009-10-08 00:00:00    3         3000

100  2009-10-08 00:00:00    3         1500

101  2009-11-20 00:00:00    2         1560

103  2008-05-20 00:00:00    4         2060
```

```
SQL> SELECT ID, NAME, AGE, AMOUNT
  2  FROM CUSTOMERS, ORDERS
  3  WHERE  CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

```
ID NAME      AGE   AMOUNT
-- -----     ---   ------
3  kaushik   23    3000
3  kaushik   23    1500
2  Khilan    25    1560
4  Chaitali  25    2060
```

```
SQL> SELECT  ID, NAME, AMOUNT, DATE
  2  FROM CUSTOMERS
  3  INNER JOIN ORDERS
  4  ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

```
ID   NAME      AMOUNT   DATE
---  ------    -------- ------
3   kaushik    3000    2009-10-08 00:00:00
3   kaushik    1500    2009-10-08 00:00:00
2   Khilan     1560    2009-11-20 00:00:00
4   Chaitali   2060    2008-05-20 00:00:00
```

```
SQL> SELECT  ID, NAME, AMOUNT, DATE
  2  FROM CUSTOMERS
  3  LEFT JOIN ORDERS
```

```
  4  ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;


ID  NAME       AMOUNT DATE
--  ------     ------ -------
1   Ramesh      NULL   NULL
2   Khilan      1560   2009-11-20 00:00:00
3   kaushik     3000   2009-10-08 00:00:00
3   kaushik     1500   2009-10-08 00:00:00
4   Chaitali    2060   2008-05-20 00:00:00
5   Hardik      NULL   NULL
6   Komal       NULL   NULL
7   Muffy       NULL   NULL

SQL> SELECT  ID, NAME, AMOUNT, DATE
  2  FROM CUSTOMERS
  3  RIGHT JOIN ORDERS
  4  ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;


ID  NAME       AMOUNT DATE
--  -------    ------ ---------
3   kaushik     3000   2009-10-08 00:00:00
3   kaushik     1500   2009-10-08 00:00:00
```

```
2   Khilan      1560   2009-11-20 00:00:00

4   Chaitali    2060   2008-05-20 00:00:00


SQL> SELECT  ID, NAME, AMOUNT, DATE

  2  FROM CUSTOMERS

  3  FULL JOIN ORDERS

  4  ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;



ID NAME       AMOUNT DATE

-- -----      ------ ----------

1  Ramesh     NULL  NULL

2  Khilan     1560  2009-11-20 00:00:00

3  kaushik    3000  2009-10-08 00:00:00

3  kaushik    1500  2009-10-08 00:00:00

4  Chaitali   2060  2008-05-20 00:00:00

5  Hardik     NULL  NULL

6  Komal      NULL  NULL

7  Muffy      NULL  NULL

3  kaushik    3000  2009-10-08 00:00:00

3  kaushik    1500  2009-10-08 00:00:00

2  Khilan     1560  2009-11-20 00:00:00
```

4 Chaitali 2060 2008-05-20 00:00:00

EXP 9

AIM: To implement triggers.

Theory:

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events:

- A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).
- A database definition (DDL) statement (CREATE, ALTER, or DROP).
- A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers could be defined on the table, view, schema, or database with which the event is associated.

Benefits of Triggers

Triggers can be written for the following purposes:

- Generating some derived column values automatically
- Enforcing referential integrity
- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables
- Imposing security authorizations
- Preventing invalid transactions

Creating Triggers

The syntax for creating a trigger is:

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
```

```
DECLARE
   Declaration-statements
BEGIN
   Executable-statements
EXCEPTION
   Exception-handling-statements
END;
```

Where,

- CREATE [OR REPLACE] TRIGGER trigger_name: Creates or replaces an existing trigger with the *trigger_name*.
- {BEFORE | AFTER | INSTEAD OF} : This specifies when the trigger would be executed. The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE}: This specifies the DML operation.
- [OF col_name]: This specifies the column name that would be updated.
- [ON table_name]: This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n]: This allows you to refer new and old values for various DML statements, like INSERT, UPDATE, and DELETE.
- [FOR EACH ROW]: This specifies a row level trigger, i.e., the trigger would be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
- WHEN (condition): This provides a condition for rows for which the trigger would fire. This clause is valid only for row level triggers.

Example:

To start with, we will be using the CUSTOMERS table we had created and used in the previous chapters:

```
Select * from customers;
```

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
+----+----------+-----+-----------+----------+
```

The following program creates a **row level** trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new values:

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
```

```
DECLARE
   sal_diff number;
BEGIN
   sal_diff := :NEW.salary  - :OLD.salary;
   dbms_output.put_line('Old salary: ' || :OLD.salary);
   dbms_output.put_line('New salary: ' || :NEW.salary);
   dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

When the above code is executed at SQL prompt, it produces the following result:

```
Trigger created.
```

Here following two points are important and should be noted carefully:

- OLD and NEW references are not available for table level triggers, rather you can use them for record level triggers.
- If you want to query the table in the same trigger, then you should use the AFTER keyword, because triggers can query the table or change it again only after the initial changes are applied and the table is back in a consistent state.
- Above trigger has been written in such a way that it will fire before any DELETE or INSERT or UPDATE operation on the table, but you can write your trigger on a single or multiple operations, for example BEFORE DELETE, which will fire whenever a record will be deleted using DELETE operation on the table.

EXP 10

AIM : Case study on Hospital Management System

Explain with proper E-R diagram.