DATA BASE MANAGEMENT SYSTEM

LABORATORY MANUAL

DIPLOMA

(II YEAR –IV SEM)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJA JAIT SINGH COLLEGE OF DIPLOMA AND ENGINEERING
(Govt. of India)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION

- To improve the quality of technical education that provides efficient software engineers with an attitude to adapt challenging IT needs of local, national and International arena, through teaching and interaction with alumni and industry.

MISSION

- Department intends to meet the contemporary challenges in the field of IT and is playing a vital role in shaping the education of the 21st century by providing unique Educational and research opportunities.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1 – ANALYTICAL SKILLS

To facilitate the graduates with the ability to visualize, gather information, articulate, analyse, solve complex problems, and make decisions. These are essential to address the challenges of complex and computation intensive problems increasing their productivity.

PEO2 – TECHNICAL SKILLS

To facilitate the graduates with the technical skills that prepare them for immediate employment and pursue certification providing a deeper understanding of the technology in advanced areas of computer science and related fields, thus encouraging to pursue higher education and research based on their interest.

PEO3 – SOFT SKILLS

To facilitate the graduates with the soft skills that include fulfilling the mission, setting goals, showing self-confidence by communicating effectively, having a positive attitude, get involved in team-work, being a leader, managing their career and their life.

PEO4 – PROFESSIONAL ETHICS

To facilitate the graduates with the knowledge of professional and ethical responsibilities by paying attention to grooming, being conservative with style, following dress codes, safety codes, and adapting themselves to technological advancements.

PROGRAM SPECIFIC OUTCOMES (PSOs)

After the completion of the course, B. Tech Information Technology, the graduates will have the following Program Specific Outcomes:

1. **Fundamentals and critical knowledge of the Computer System:-** Able to Understand the working principles of the computer System and its components , Apply the knowledge to build, asses, and analyse the software and hardware aspects of it .

2. **The comprehensive and Applicative knowledge of Software Development:** Comprehensive skills of Programming Languages, Software process models, methodologies, and able to plan, develop, test, analyse, and manage the software and hardware intensive systems in heterogeneous platforms individually or working in teams.

3. **Applications of Computing Domain & Research:** Able to use the professional, managerial, interdisciplinary skill set, and domain specific tools in development processes, identify the research gaps, and provide innovative solutions to them.

List of Practicals

S.no	Topic name	Date	Sign
1	**Exercises on creation and modification of structure of tables.**		
2	**Exercises on inserting and deleting values from tables.**		
3	**Exercises on querying the table (using select command).**		
4	**Exercises on using various types of joins.**		
5	**Exercises on using functions provided by database package**		
6	**Exercises on commands like Grant, Revoke, Commit and Rollback etc.**		
7	**Exercise on INDEX**		
8	**Exercise a query to use unique, primary key and check constraints**		
9	**Exercise a Query to execute different operators in SQL**		

II Year DIPLOMA CSE - IV Semester **L T/P**

-3 /3

DATA BASE MANAGEMENT SYSTEMS LAB

OBJECTIVES:

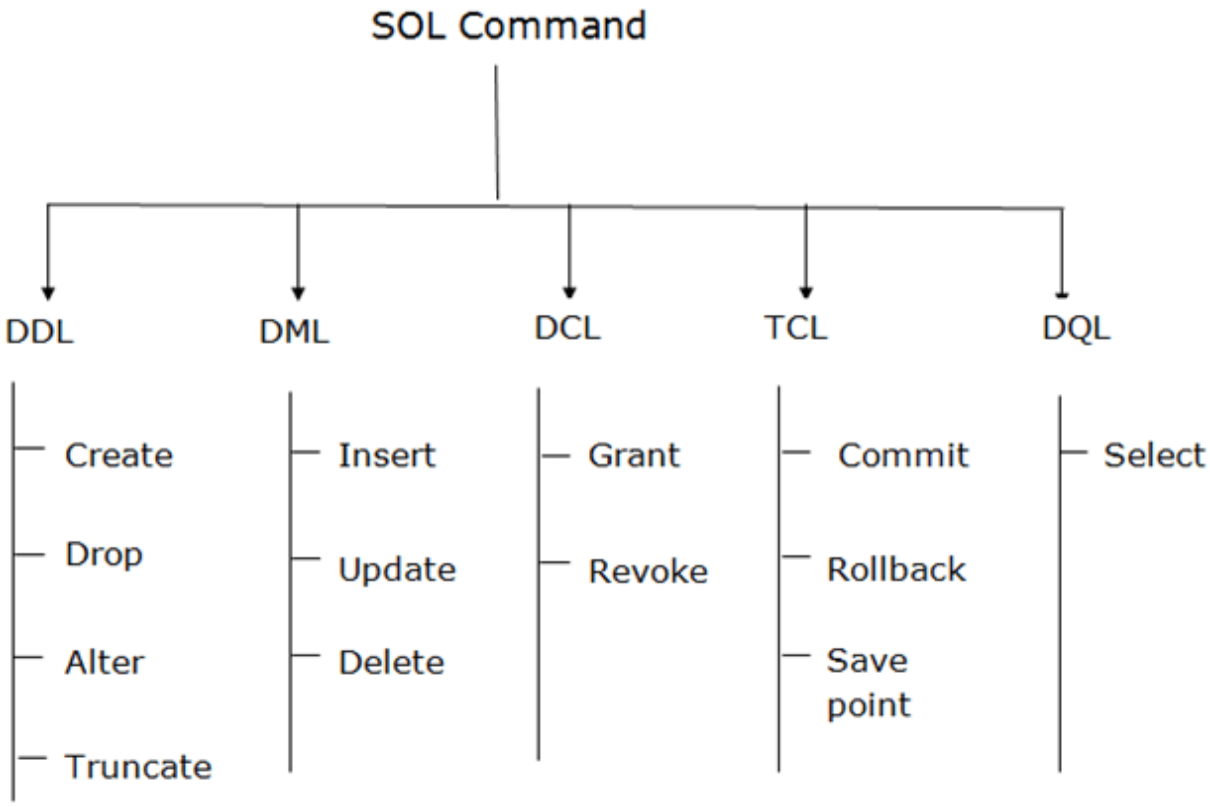
- To present an introduction to database management systems, with an emphasis on how to organize, maintain and retrieve - efficiently, and effectively - information from a DBMS.
- Identify the purpose of different Data Base management software applications.

Structure Query Language(SQL)

Structure Query Language(SQL) is a database query language used for storing and managing data in Relational DBMS. SQL was the first commercial language introduced for E.F Codd's Relational model of database. Today almost all RDBMS(MySql, Oracle, Infomix, Sybase, MS Access) use SQL as the standard database query language. SQL is used to perform all types of data operations in RDBMS.

Types of SQL Commands

There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.


SOL Command
DDL
DML
DCL
TCL
DQL
Create
Drop
Alter
Truncate
Insert
Update
Delete
Grant
Revoke
Commit
Rollback
Save point
Select

EXPERIMENT – I

1. Exercises on creation and modification of structure of tables.

DDL commands:

- CREATE – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
- DROP – is used to delete objects from the database.
- ALTER-is used to alter the structure of the database.
- TRUNCATE–is used to remove all records from a table, including all spaces allocated for the records are removed.
- RENAME –is used to rename an object existing in the database.

Exercises based on commands:

EmployeeID	EmployeeName	Emergency ContactName	PhoneNumber	Address	City	Country
01	Shanaya	Abhinay	9898765612	Oberoi Street 23	Mumbai	India
02	Anay	Soumya	9432156783	Marathalli House No 23	Delhi	India
03	Preeti	Rohan	9764234519	Queens Road 45	Bangalore	India
04	Vihaan	Akriti	9966442211	Brigade Road Block 4	Hyderabad	India
05	Manasa	Shourya	9543176246	Mayo Road 23	Kolkata	India

CREATE

This statement is used to create a table or a database.

The 'CREATE DATABASE' Statement

As the name suggests, this statement is used to create a database.

Syntax

CREATE DATABASE DatabaseName;

Example

CREATE DATABASE Employee;

The 'CREATE TABLE' Statement

This statement is used to create a table.

Syntax

CREATE TABLE TableName (Column1datatype,Column2datatype,Column3datatype, ColumnN datatype);

CREATE TABLE Employee_Info (EmployeeID int, EmployeeName varchar(255), Emergency ContactName varchar(255), PhoneNumber int, Address varchar(255), City varchar(255), Country varchar(255));

Example

We can also create a table using another table. Refer the below sytax and example:

The 'CREATE TABLE AS' Statement

Syntax

CREATE TABLE NewTableName AS SELECT Column1, column2,....., ColumnN FROM ExistingTableName WHERE.....;

Example

CREATE TABLE ExampleTable AS SELECT EmployeeName, PhoneNumber FROM Employee_Info;

DROP

This statement is used to drop an existing table or a database.

The 'DROP DATABASE' Statement

This statement is used to drop an existing database. When you use this statement, complete information present in the database will be lost.

Syntax

DROP DATABASE DatabaseName;

Example

DROP DATABASE Employee;

The 'DROP TABLE' Statement

This statement is used to drop an existing table. When you use this statement, complete information present in the table will be lost.

Syntax

DROP TABLE TableName;
Example

DROP Table Employee_Info;

TRUNCATE

This command is used to delete the information present in the table but does not delete the table. So, once you use this command, your information will be lost, but not the table.

Syntax

TRUNCATE TABLE TableName;

Example

TRUNCATE Table Employee_Info;

ALTER

This command is used to delete, modify or add constraints or columns in an existing table.

The 'ALTER TABLE' Statement

This statement is used to add, delete, modify columns in an existing table.

The 'ALTER TABLE' Statement with ADD/DROP COLUMN

You can use the ALTER TABLE statement with ADD/DROP Column command according to your need. If you wish to add a column, then you will use the ADD command, and if you wish to delete a column, then you will use the DROP COLUMN command.

Syntax
ALTER TABLE TableName ADD
ColumnName Datatype;

ALTER TABLE TableName DROP
COLUMN ColumnName;
Example

--ADD Column BloodGroup:

ALTER TABLE Employee_Info ADD BloodGroup varchar(255);

--DROP Column BloodGroup:

ALTER TABLE Employee_Info DROP COLUMN BloodGroup ;

The 'ALTER TABLE' Statement with ALTER/MODIFY COLUMN

This statement is used to change the datatype of an existing column in a table.

Syntax

ALTER TABLE TableName
ALTER COLUMN ColumnName Datatype;
Example

--Add a column DOB and change the data type to Date.

ALTER TABLE Employee_Info ADD DOB year;

ALTER TABLE Employee_Info
ALTER DOB date;

2. Exercises on inserting and deleting values from tables.

Data Manipulation Language (DML Commands in SQL)

Data Manipulation Language is used to manipulate data within the tables. The basic DML commands in SQL are Insert, Update and Delete.

SELECT	Retrieve information from database
INSERT	Add new information to a database
UPDATE	Modifies the information currently stored in a database
DELETE	Delete information from the database

Data Manipulation Language Commands (DML)

The commands are as follows:

- INSERT INTO
- UPDATE
- DELETE
- SELECT

Apart from these commands, there are also other manipulative operators/functions such as:

Aggregate Functions

- NULL Functions

USE

-

The USE statement is used to select the database on which you want to perform operations.

Syntax

USE DatabaseName;

Example

USE Employee;

INSERT INTO

This statement is used to insert new records into the table.

Syntax

INSERT INTO TableName (Column1, Column2, Column3, ...,ColumnN) VALUES (value1, value2, value3, ...);

--If you don't want to mention the column names then use the below syntax INSERT INTO TableName VALUES (Value1, Value2, Value3, ...);

Example

INSERT INTO Employee_Info(EmployeeID, EmployeeName, Emergency ContactName, PhoneNumber, Address, City, Country) VALUES ('06', 'Sanjana','Jagannath', '9921321141', 'Camel Street House No 12', 'Chennai', 'India');

INSERT INTO Employee_InfoVALUES ('07', 'Sayantini','Praveen', '9934567654', 'Nice Road 21', 'Pune', 'India');

Example UPDATE

This statement is used to modify the records already present in the table.

Syntax

UPDATE TableName
SET Column1 = Value1, Column2 = Value2, ... WHERE Condition;

Example

UPDATE Employee_Info SET EmployeeName = 'Aahana', City= 'Ahmedabad' WHERE EmployeeID = 1;

DELETE

This statement is used to delete the existing records in a table.

Syntax

DELETE FROM TableName WHERE Condition;

Example

DELETE FROM Employee_Info WHERE EmployeeName='Preeti';

2. Exercises on querying the table (using select command). Data Query Language

DQL is used to fetch the data from the database.

It uses only one command:

- SELECT

a. SELECT: This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

Syntax:

1. SELECT expressions FROM TABLES WHERE conditions; For example:

3. SELECT emp_name FROM employee WHERE age > 20;

SELECT

This statement is used to select data from a database and the data returned is stored in a result table, called the **result-set**.

Syntax

```
SELECT Column1, Column2, ...ColumN FROM
TableName;

--(*) is used to select all from the table
SELECT * FROM table_name;

-- To select the number of records to return use:
SELECT TOP 3 * FROM TableName;
```

Example

```
SELECT EmployeeID, EmployeeName FROM Employee_Info;

--(*) is used to select all from the table

SELECT * FROM Employee_Info;


-- To select the number of records to return use:
SELECT TOP 3 * FROM Employee_Info;
```

Apart from just using the SELECT keyword individually, you can use the following keywords with the SELECT statement:

- DISTINCT
- ORDER BY
- GROUP BY
- HAVING Clause
- INTO

The 'SELECT DISTINCT' Statement

This statement is used to return only different values.

Syntax

SELECT DISTINCT Column1, Column2, ...ColumnN FROM
TableName;
SELECT DISTINCT PhoneNumber FROM Employee_Info;
Example

The 'ORDER BY' Statement

The 'ORDER BY' statement is used to sort the required results in ascending or descending order. The results are sorted in ascending order by default. Yet, if you wish to get the required results in descending order, you have to use the **DESC** keyword.

Syntax

SELECT Column1, Column2, ...ColumnN

FROM TableName
ORDER BY Column1, Column2, ... ASC|DESC;

Example

SELECT * FROM Employee_Info ORDER BY EmergencyContactName DESC;

-- Select all employees from the 'Employee_Info' table sorted by EmergencyContactName and EmployeeName:

SELECT * FROM Employee_Info ORDER BY EmergencyContactName, EmployeeName;

/* Select all employees from the 'Employee_Info' table sorted by EmergencyContactName in Descending order and EmployeeName in Ascending order: */

SELECT * FROM Employee_Info ORDER BY EmergencyContactName ASC, EmployeeName DESC;

The 'GROUP BY' Statement

This 'GROUP BY' statement is used with the aggregate functions to group the result-set by one or more columns.

Syntax

SELECT Column1, Column2,..., ColumnN FROM
TableName
WHERE Condition
GROUP BY ColumnName(s) ORDER
BY ColumnName(s);

SELECT COUNT(EmployeeID), City FROM Employee_Info GROUP BY City;

To list the number of employees from each city.

Example

The 'HAVING' Clause

The 'HAVING' clause is used in SQL because the **WHERE keyword** cannot be used everywhere.

Syntax

SELECT ColumnName(s) FROM
TableName WHERE Condition
GROUP BY ColumnName(s) HAVING
Condition
ORDER BY ColumnName(s);

Example

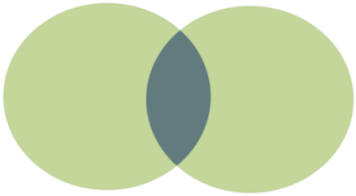
/* To list the number of employees in each city. The employees should be sorted high to low and only those cities must be
ncluded who have more than 5 employees:*/

SELECT COUNT(EmployeeID), City FROM Employee_Info GROUP BY City HAVING
COUNT(EmployeeID) > 2 ORDER BY COUNT(EmployeeID) DESC;

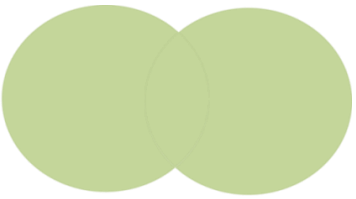
4. Exercises on using various types of joins.

JOINS are used to combine rows from two or more tables, based on a related column between those tables. The following are the types of joins:

- **INNER JOIN:** This join returns those records which have matching values in both the tables.
- **FULL JOIN:** This join returns all those records which either have a match in the left or the right table.
- **LEFT JOIN:** This join returns records from the left table, and also those records which satisfy the condition from the right table.
- **RIGHT JOIN:** This join returns records from the right table, and also those records which satisfy the condition from the left table.



INNER JOIN



FULL JOIN

LEFT JOIN

RIGHT JOIN

TechID	EmpID	TechName	ProjectStartDate
1	10	DevOps	04-01-2019
2	11	Blockchain	06-07-2019
3	12	Python	01-03-2019

INNER JOIN

Syntax

SELECT ColumnName(s) FROM
Table1
INNER JOIN Table2 ON Table1.ColumnName = Table2.ColumnName;

SELECT Technologies.TechID, Employee_Info.EmployeeName FROM Technologies INNER JOIN Employee_Info ON Technologies.EmpID = Employee_Info.EmpID;

FULL JOIN

Syntax

SELECT ColumnName(s) FROM
Table1
FULL OUTER JOIN Table2 ON Table1.ColumnName =
Table2.ColumnName;

Example

SELECT Employee_Info.EmployeeName, Technologies.TechID FROM Employee_Info FULL OUTER JOIN Orders ON Employee_Info.EmpID=Employee_Salary.EmpID ORDER BY Employee_Info.EmployeeName;

LEFT JOIN

Syntax

SELECT ColumnName(s) FROM
Table1
LEFT JOIN Table2 ON Table1.ColumnName = Table2.ColumnName;

SELECT Employee_Info.EmployeeName, Technologies.TechID
FROM Employee_Info LEFT JOIN Technologies ON Employee_Info.EmployeeID = Technologies.EmpIDID ORDER BY Employee_Info.EmployeeName;

RIGHT JOIN

Syntax

SELECT ColumnName(s) FROM
Table1
RIGHT JOIN Table2 ON Table1.ColumnName = Table2.ColumnName;

Example

```
SELECT Technologies.TechID
FROM Technologies
RIGHT JOIN Employee_Info ON Technologies.EmpID = Employee_Info.EmployeeID
ORDER BY Technologies.TechID;
```

5. Exercises on using functions provided by database package.

This section of the article will include the following functions:

- MIN()
- MAX()
- COUNT()
- SUM()
- AVG()

MIN() Function

The MIN function returns the smallest value of the selected column in a table.

Syntax

```
SELECT MIN(ColumnName) FROM
TableName
WHERE Condition;
```

Example

```
SELECT MIN(EmployeeID) AS SmallestID FROM Employee_Info;
```

MAX() Function

The MAX function returns the largest value of the selected column in a table.

Syntax

```
SELECT MAX(ColumnName)
FROM TableName
WHERE Condition;
```

Example

```
SELECT MAX(Salary) AS LargestFees FROM Employee_Salary;
```

COUNT() Function

The COUNT function returns the number of rows which match the specified criteria.

Syntax

```
SELECT COUNT(ColumnName) FROM
TableName
WHERE Condition;
```

Example

```
SELECT COUNT(EmployeeID) FROM Employee_Info;
```

SUM() Function

The SUM function returns the total sum of a numeric column that you choose.

Syntax

SELECT SUM(ColumnName) FROM
TableName
WHERE Condition;
Example

SELECT SUM(Salary) FROM Employee_Salary;

AVG() Function

The AVG function returns the average value of a numeric column that you choose.

Syntax

SELECT AVG(ColumnName) FROM
TableName
WHERE Condition;

Example

SELECT AVG(Salary) FROM Employee_Salary;

NULL Functions

The NULL functions are those functions which let you return an alternative value if an expression is NULL. In the SQL Server, the function is **ISNULL()**.

```
SELECT EmployeeID * (Month_Year_of_Salary + ISNULL(Salary, 0)) FROM

Employee_Salary;
```

6. Exercises on commands like Grant, Revoke, Commit and Rollback etc.

Data Control Language Commands (DCL)

DCL commands are used to grant and take back authority from any database user.

Commands that come under DCL:

- Grant
- Revoke
- Commit

a.Grant: It is used to give user access privileges to a database.
b. **Revoke:** It is used to take back permissions from the user.

GRANT

This command is used to provide access or privileges on the database and its objects to the users.

Syntax

```
GRANT PrivilegeName ON
ObjectName
TO {UserName |PUBLIC |RoleName} [WITH
GRANT OPTION];
```

where,

- **PrivilegeName** – Is the privilege/right/access granted to the user.
- **ObjectName** – Name of a database object like TABLE/VIEW/STORED PROC.
- **UserName** – Name of the user who is given the access/rights/privileges.
- **PUBLIC** – To grant access rights to all users.
- **RoleName** – The name of a set of privileges grouped together.
- **WITH GRANT OPTION** – To give the user access to grant other users with rights.

Example

```
To grant SELECT permission to Employee_Info table to user1
GRANT SELECT ON Employee_Info TO user1;
```

REVOKE

This command is used to withdraw the user's access privileges given by using the GRANT command.

Syntax

```
REVOKE PrivilegeName ON
ObjectName
FROM {UserName |PUBLIC |RoleName}
```

Example

```
-- To revoke the granted permission from user1
REVOKE SELECT ON Employee_Info TO user1;
```

Transaction Control Language

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Commands that come under TCL:

- COMMIT
- ROLLBACK
- SAVEPOINT

a. Commit: Commit command is used to save all the transactions to the database.

Syntax:

1. COMMIT;

Example:

1. DELETE FROM CUSTOMERS
2. WHERE AGE = 25;
3. COMMIT;

b. Rollback: Rollback command is used to undo transactions that have not already been saved to the database.

Syntax:

1.

ROLLBA

CK;

Example:

1. DELETE FROM CUSTOMERS
2. WHERE AGE = 25;
3. ROLLBACK;

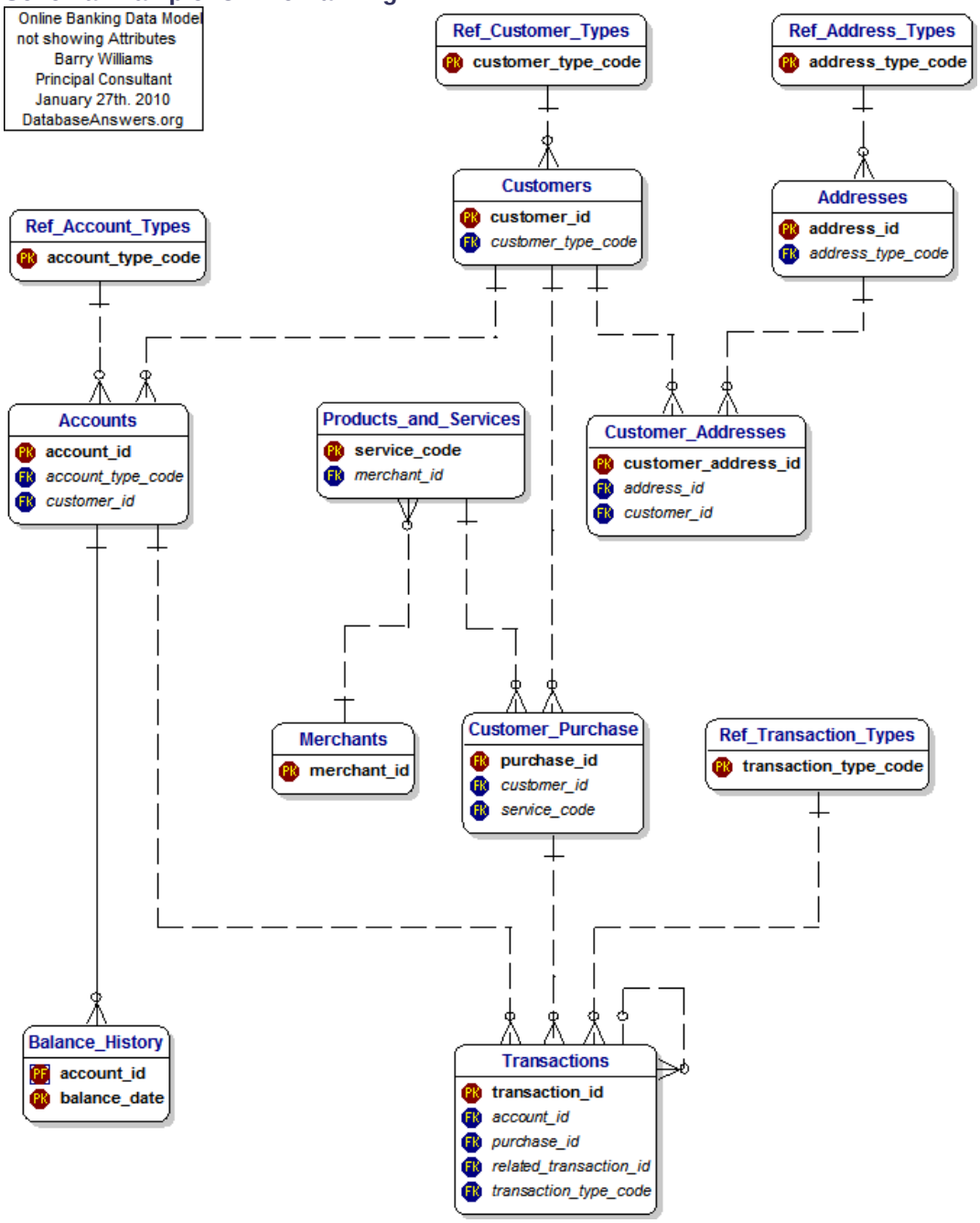
c. SAVEPOINT: It is used to roll the transaction back to a certainpoint without rolling back the entire transaction.

Syntax:

1. SAVEPOINT SAVEPOINT_NAME;

6. Design of database for any application.

Schema Example: Online Banking



Image

```
CREATE DATABASE
-- Table structure for table `account_customers`
DROP TABLE IF EXISTS `account_customers`;
CREATE TABLE `account_customers` (
  `Account_id` int(10) unsigned NOT NULL,
  `Customer_id` int(10) unsigned NOT NULL,
  PRIMARY KEY (`Customer_id`,`Account_id`),
  KEY `fk_Accounts (`Customer_id`),
  KEY `fk_Accounts1_idx` (`Account_id`);
```

8. **Exercise on INDEX.**

INDEX

This constraint is used to create indexes in the table, through which you can create and retrieve data from the database very quickly.

Syntax

```
--Create an Index where duplicate values are allowed
CREATE INDEX IndexName
ON TableName (Column1, Column2, ...ColumnN);

--Create an Index where duplicate values are not allowed
CREATE UNIQUE INDEX IndexName
ON TableName (Column1, Column2, ...ColumnN);
```

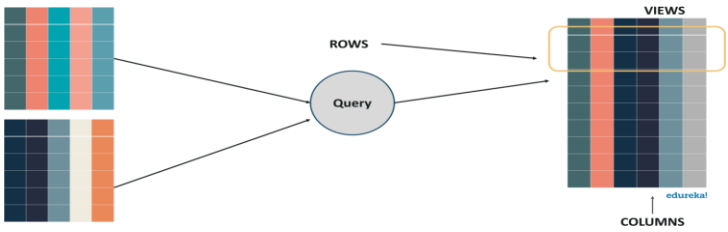
Example

```
CREATE INDEX idex_EmployeeName ON Persons (EmployeeName);

--To delete an index in a table

DROP INDEX Employee_Info.idex_EmployeeName;
```

10.Use of views with examples.

SQL Commands: Views

A view in SQL is a single table, which is derived from other tables. So, a view contains rows and columns similar to a real table and has fields from one or more table.



The 'CREATE VIEW' statement

This statement is used to create a view, from a table.

Syntax

```
CREATE VIEW ViewName AS
SELECT Column1, Column2, ..., ColumnN
FROM TableName
WHERE Condition;
```

Example

```
CREATE VIEW [Kolkata
Employees] AS SELECT
EmployeeName,
PhoneNumber FROM
Employee_Info
WHERE City = "Kolkata";
```

The 'CREATE OR REPLACE VIEW' statement

This statement is used to update a view.

Syntax

```
CREATE VIEW OR REPLACE ViewName AS
SELECT Column1, Column2, ..., ColumnN
FROM TableName
WHERE Condition;
```

Example

```
CREATE VIEW OR REPLACE [Kolkata Employees] AS SELECT
EmployeeName, PhoneNumber
FROM Employee_Info WHERE City = "Kolkata";
```

The 'DROP VIEW' statement

This statement is used to delete a view.

Syntax

```
DROP VIEW ViewName;
```

Example

```
DROP VIEW [Kolkata Employees];
```

execute different operators in SQL

8 .Exercise a query to use unique, primary key and check constraints

SQL UNIQUE Constraint

The UNIQUE constraint ensures that all values in a column are different.

Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.

A PRIMARY KEY constraint automatically has a UNIQUE constraint.

However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

Exercise a query to use unique, primary key and check constraints

SQL UNIQUE Constraint on CREATE TABLE

The following SQL creates a UNIQUE constraint on the "ID" column when the "Persons" table is created:

SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons (
    ID int NOT NULL UNIQUE,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

MySQL:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
```

```
    FirstName varchar(255),
    Age int,
    UNIQUE (ID)
);
```

SQL UNIQUE Constraint on ALTER TABLE

To create a UNIQUE constraint on the "ID" column when the table is already created, use the following SQL:

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ADD UNIQUE (ID);
```

To name a UNIQUE constraint, and to define a UNIQUE constraint on multiple columns, use the following SQL syntax:

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ADD CONSTRAINT UC_Person UNIQUE (ID,LastName);
```

DROP a UNIQUE Constraint

To drop a UNIQUE constraint, use the following SQL:

MySQL:

```
ALTER TABLE Persons
DROP INDEX UC_Person;
```

SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
DROP CONSTRAINT UC_Person;
```

SQL PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

SQL PRIMARY KEY on CREATE TABLE

The following SQL creates a PRIMARY KEY on the "ID" column when the "Persons" table is created:

MySQL:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
```

```
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
);
```

SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons (
    ID int NOT NULL PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

MySQL / SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)
);
```

SQL PRIMARY KEY on ALTER TABLE

To create a PRIMARY KEY constraint on the "ID" column when the table is already created, use the following SQL:

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ADD CONSTRAINT PK_Person PRIMARY KEY (ID,LastName);
```

Note: If you use ALTER TABLE to add a primary key, the primary key column(s) must have been declared to not contain NULL values (when the table was first created).

DROP a PRIMARY KEY Constraint

To drop a PRIMARY KEY constraint, use the following SQL:

MySQL:

```
ALTER TABLE Persons
DROP PRIMARY KEY;
```

SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
DROP CONSTRAINT PK_Person;
```

SQL CHECK Constraint

The CHECK constraint is used to limit the value range that can be placed in a column.

If you define a CHECK constraint on a column it will allow only certain values for this column.

If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

SQL CHECK on CREATE TABLE

The following SQL creates a CHECK constraint on the "Age" column when the "Persons" table is created. The CHECK constraint ensures that the age of a person must be 18, or older:

MySQL:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
);
```

SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int CHECK (Age>=18)
);
```

To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns, use the following SQL syntax:

MySQL / SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255),
    CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Sandnes')
);
```

SQL CHECK on ALTER TABLE

To create a CHECK constraint on the "Age" column when the table is already created, use the following SQL:

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
```

```
ADD CHECK (Age>=18);
```

To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns, use the following SQL syntax:

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ADD CONSTRAINT CHK_PersonAge CHECK (Age>=18 AND City='Sandnes');
```

DROP a CHECK Constraint

To drop a CHECK constraint, use the following SQL:

SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
DROP CONSTRAINT CHK_PersonAge;
```

MySQL:

```
ALTER TABLE Persons
DROP CHECK CHK_PersonAge;
```

9 Exercise a Query to execute different operators in SQL

SQL Arithmetic Operators

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

SQL Bitwise Operators

Operator	Description
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR

SQL Comparison Operators

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

SQL Compound Operators

Operator	Description

+=	Add equals
-=	Subtract equals
*=	Multiply equals
/=	Divide equals
%=	Modulo equals
&=	Bitwise AND equals
^-=	Bitwise exclusive equals
|*=	Bitwise OR equals

SQL Logical Operators

Operator	**Description**
ALL	TRUE if all of the subquery values meet the condition
AND	TRUE if all the conditions separated by AND is TRUE
ANY	TRUE if any of the subquery values meet the condition
BETWEEN	TRUE if the operand is within the range of comparisons

EXISTS	TRUE if the subquery returns one or more records
IN	TRUE if the operand is equal to one of a list of expressions
LIKE	TRUE if the operand matches a pattern
NOT	Displays a record if the condition(s) is NOT TRUE
OR	TRUE if any of the conditions separated by OR is TRUE
SOME	TRUE if any of the subquery values meet the condition