DATABASE MANAGEMENT SYSTEM

* Database: Database is a collection of logically related data that is stored in order to meet requirements of the user. It can be shared, accessed, managed and updated. For example:

  Names, Telephone numbers & addresses of people.

* Database Management System (DBMS): It is a set of programs that controls the creation, storage, management and retrieval of data from database. It is indirectly called as a software (set of programs).

* Different types of DBMS software:

  i) mySQL ii) MS Access iii) Oracle iv) DB2, DBase, ...

* Applications of DBMS: Banking, Institutions, Company, Airlines, Universities, Telecommunications, Sales, manufacturing, HR management.

* Purpose of DBMS systems: In early days, database applications were build on the top of file systems. Due to the drawbacks of using file systems such as:

  i) Data redundancy/Inconsistency

Data redundancy cause: multiple file formats, duplication of information in different files.

ii) Difficulty in accessing data

16/3

* Disadvantage of Files:

iii) Data integrity problems: The data present in file should be consistent & correct, and inorder to achieve this, the data must satisfy certain constraints like account balance > 0 which become part of the program code and it becomes hard to add new constraints or modify the existing ones.

* Atomicity of updates: Failures may leave database in an inconsistent state with partial updates being carried out. Ex: Transfer of funds from one account to another should either complete or not happen at all.

* Unauthorised access: Anyone who gets access to a file can read/modify the data.

* Problems in concurrent access:

When a number of users operate on common data at the same time, then problems arise due to the lack of concurrency control.

* ADVANTAGES OF DATABASE SYSTEMS:

i) Reduced Redundancy

ii) Data consistency

iii) Improved security

iv) Data access is efficient

v) concurrency control, etc.

* Differentiate between file systems & Database systems.

→ Database Languages: These are used to perform read, update and store data in your databases.
[Create, update, retrieve, Delete ⇒ CURD]

* Types of DBL: DDL, DML, DCL, DQL

[Data is represented in forms of tables in DBMS].

* DDL: DDL is used for specifying the database schema. Lets take SQL for instance to categorize the statements that come under DDL (Data Definition Language):-

i) To create a database instance we use keyword 'CREATE'.

ii) To alter the structure of database, we use 'ALTER'.

iii) To drop database instance, we use 'DROP'

iv) To ~~remove~~ rename database instances, we use 'RENAME'.

→ Creation of a Database: CREATE DATABASE DatabaseName;

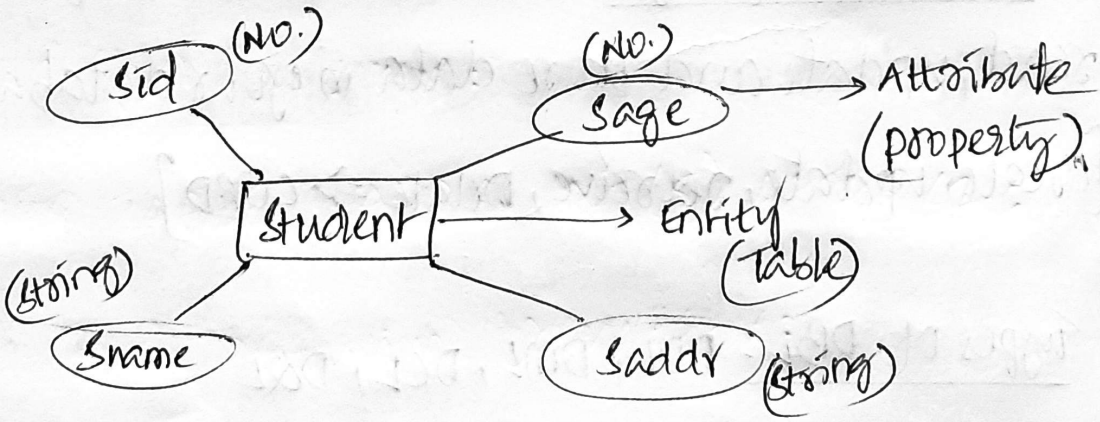
For ex: CREATE DATABASE CSEB; (Enter next)

Next step: USE CSEB; (selecting this database)

→ Creation of a Table: CREATE TABLE TableName (

List of attributes with its datatypes,);

Eg:



[Entity = Student, no. of attributes = 4]

Eg: CREATE TABLE student (

Sid INT(2),

Sname VARCHAR(20),

Sage INT(2),

Saddr VARCHAR(100)

);

DESC student;

(OR) DESCRIBE student;

⟹

Sid	Sname	Sage	Saddr

* Create a table 'product' with PId, Pname, Pquantity, Pcost and Pcompany as its attributes by checking whether the table already exists.

A) CREATE TABLE IF NOT EXISTS product (

PId INT(5), Pname VARCHAR(50),

Pquantity INT(3), Pcost DECIMAL(6),

Pcompany VARCHAR(30));

21/3

* Points to Remember while working with Table:

i) Name of the tables in the database should be

ii) unique and columns in the table should not be duplicated.

iii) You have to specify the datatypes for attributes (or columns) while creating a table.

→ ALTER: Alter command is applied to modify the structure of the current tables that are present in the database, we can perform addition, deletion and altering the columns of table.

It takes the following form:

ALTER TABLE tableName add columnName Datatype;

Eg: CREATE TABLE stud (sid INT(5));

ALTER TABLE stud ADD sname VARCHAR(20);

* To add a column at beginning:

ALTER TABLE T1 ADD C4 INT(5) FIRST;

To add after a column:

ALTER TABLE T1 ADD C4 INT(5) AFTER C2;

Inorder to drop column from table:

ALTER TABLE T1 DROP C4;

→ modifying the existing column with respect to its data type:

ALTER TABLE tableName MODIFY (COL) COLName Dtype;

eg: initially

CREATE TABLE stud (sid INT(5));

ALTER TABLE stud MODIFY COL sid INT(2);

* DROP: This command is used to remove the database objects, i.e; used to delete the tables that are existing in the database.

syntax: DROP TABLE TableName;

eg: DROP TABLE stud;

To remove multiple tables: DROP TABLE T1, T2, T3;

* TRUNCATE: This command is used when you want to remove all the data that is existing in the database table. syntax is:

TRUNCATE TABLE TableName;

eg: TRUNCATE TABLE stud;

* RENAME: We use this command whenever we are interested in changing the name of an existing table to a new one. Syntax is:

RENAME TABLE t1 TO t2;

Eg: RENAME TABLE stud TO student_data;

→ Q) create a Table to display the information of a student entity by taking 5 attributes; and modify the table by adding 2 more attributes and insert 5 records into the updated table.

23/3

* DML Commands: DML is used for accessing and manipulating data. The following keywords are a part of DML: 1) INSERT: It is used to insert a record / set of records into a table in a database.

It takes the following form:

INSERT INTO TableName VALUES (record);

record = set of values of attributes seperated by commas.

eg: INSERT INTO stud VALUES (001, 'Amit');

INSERT INTO stud VALUES (002, 'xyz');

→ To insert into multiple rows at once:

INSERT INTO stud (Sid, Sname) VALUES

(01, 'AAA'), (02, 'BBB'), (03, 'ccc');

ii) SELECT: In order to retrieve data from a table we use the keyword select. It's syntax is:

SELECT c1, c2, ... cn FROM TableName;

eg: To show c2 from stud(c1, c2, c3, c4, c5);

⇒ SELECT c2 FROM stud;

For more than 1 column ⇒ SELECT c1, c2 FROM stud;

For all contents ⇒ SELECT * FROM stud;

iii) UPDATE: This keyword is used whenever you're interested in updating the data that is present in the table based on certain conditions. It takes the following form:

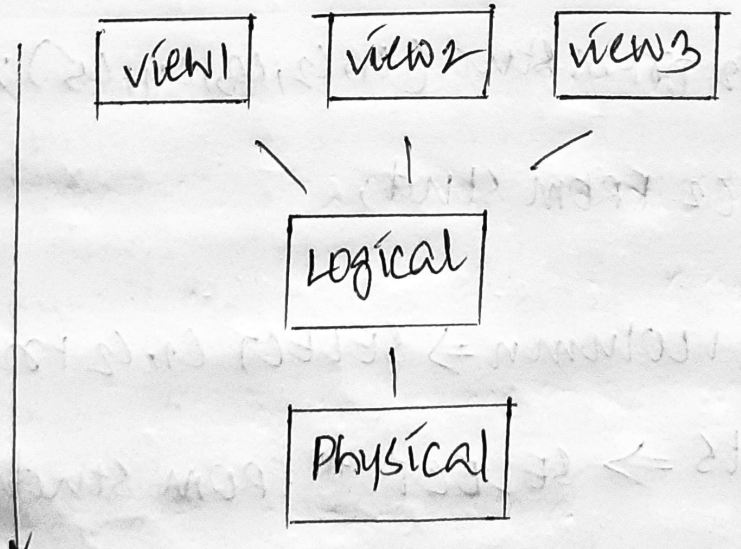
UPDATE TableName SET Col1='New' WHERE condition;

Eg: UPDATE Stud SET SId=00 WHERE Sage=20;

iv) DELETE: This keyword is used whenever you're interested in removing information from an existing table based on certain conditions.

Syntax: DELETE FROM TableName WHERE condition;

Eg: DELETE FROM Stud WHERE SId=2;



24/3

* VIEW OF DATA: Abstraction is one of the main features of database systems, i.e; hiding irrelevant details from the user and providing abstract view of data to the users helps in easy and efficient user database interaction. We will discuss 3 levels in data abstraction: namely physical level, logical level and view level.

Physical level: It is the lowest level in data abstraction and describes how actually data is stored in database. We can get an idea of complex datastructures that are used.

Logical level: Also called as conceptual level which describes what data is stored and what are the relationships that are existing among data that is present in database.

View level: Highest level of data abstraction which describes only a part of the database and also describes the user interaction with the database.

Example: Database of banking system.

View is where only customer can see data.

Logical is where person can see every customer's data.

Person in physical can see data of employee and data of costumers.

* Schema: Design of a database is called a schema and the data stored at a particular instance of time in the database is called instance of the database.

Schema:

stud (Sid, Sname, DOB);

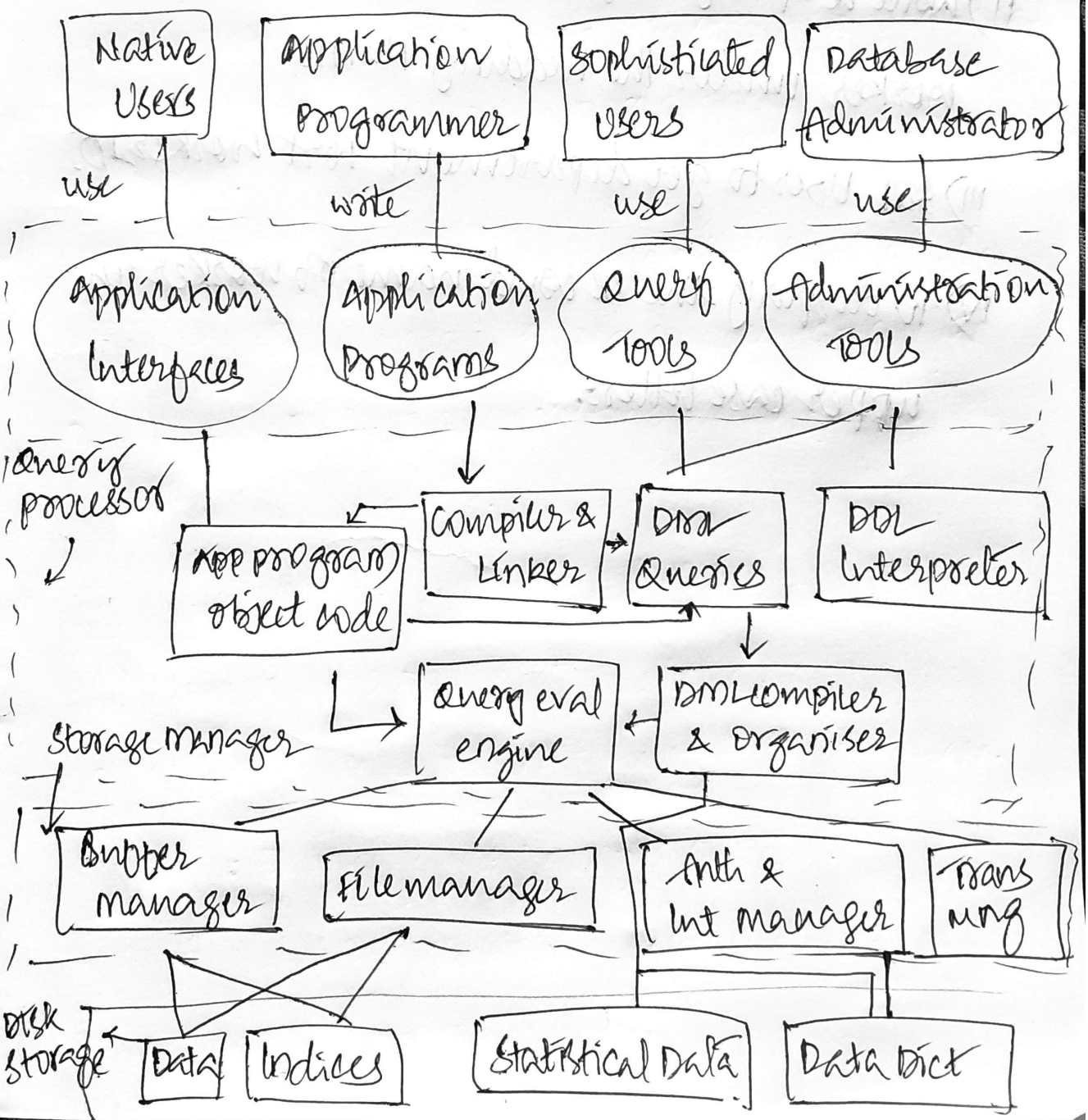
Instance:

Sid	Sname	DOB
1	Amy	3-2-21
2	John	4-8-92
3	peter	3-5-96

* Data Independence: It's usually categorised into two. First logical data independence and physical DI.

→ Logical data independence: The capacity to change the conceptual schema without having to change the external schema or any of external application programs is called as logical data independence. The changes maybe in terms of adding new entities, modifying the attributes, etc.

Physical data independence: The ability to change the internal schema without having to change conceptual schema is called as physical data independence. The changes in the internal schema may be to improve the performance of system. It may be like: Using new storage devices or modifying existing data structures, etc.



Problem: CREATE a Table called worker with WorkerID, Worker FirstName, Worker LastName, Salary, Department, Joining Dateas its attributes. Insert details of 5 different employees into the table. Answer the following queries: i) Write a query to get the firstnames of all the workers

ii) Write a query to fetch the firstname of the worker under the heading Name

iii) SQL User to get department wrt WorkerID.

iv) To display the second name of worker in upper case letters.

Get the firstnames of all the workers

ii) Write a query to fetch the firstname of the worker under the heading Name.

iii) SQL User to get department wise WorkerID.

iv) To display the second name of worker in upper case letters.

6/4

* Data Models: It is the collection of conceptual tools for describing data and data relationships.

There are different kinds of models:

i) ER model ii) Object Based Logical Models.

iii) Record Based Logical models.

* ER model: It stands for Entity Relationship model and the major components of ER Model are:

i) Rectangles ii) Ellipse iii) Diamond / Rhombus iv) Links

→ Entity: It is an object that exists and is distinguishable from other objects. It can be a thing, a place, a person, that are distinct from each other. It is represented by a rectangle with a name in it.

→ Attribute: An entity is represented by a set of attributes; the characteristics of the entity or attributes are termed as the descriptive property of an entity represented by ellipse with attribute name.

→ Types of Attributes: i) Simple: Simple are those which cannot be subdivided. Eg: Gender, Age, etc. (AGE)

ii) Composite Attribute: is one which consists of combination of more than one attribute. (OR) which can be divided. Eg: Name

iii) Single valued: It is one that has only one value for each entity. Eg: Gender

iv) Multiple Values: Attributes with more than one value. Represented by double ellipse.

Eg: (Email)

v) Derived: These are the ones which contain values that are calculated by other attributes. They are represented by dotted ellipse. Eg: If DOB is an attribute, then age is derived attribute.

vi) Key attributes: This is one which represents the key attributes of an entity. Represented by an ellipse with name of attribute underlined.

Eg: (Sid) is a key attribute.

* Relationship: It is an association that exist between one/more entities and it is represented using a diamond in ER model & we can specify the type of relationship existing bw entities by giving name.

Eg:

* Degrees: The no. of participating entities in a relationship defines the degree of that relationship.

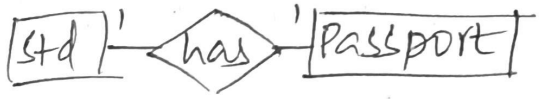
#14

* Mapping cardinalities: It describes the max no. of entities that a given entity can be associated with, via relationship. It takes the following forms:

a) ONE TO ONE (1:1) b) ONE TO MANY (1:M)

c) MANY TO ONE (N:1) d) MANY TO MANY (N:M)

→ One to One: An entity in A is associated with atmost one entity in B. And an entity in B is associated with atmost one entity in A. For ex:



→ One to many: An entity in A is associated with any no. of entities in B. And an entity in B is associated with atmost one entity in A.

→ Many to One: An entity in A is associated with atmost one entity in B. An entity in B is associated with any no. of entities in A. Eg:

→ Many TO many: An entity in A is associated with any no. of entities in B and vice versa. Eg:

(S1, S2, S3)

(C, CH, JAVA, KOTLIN)

a) Create an ER model for the following conditions:

i) College has many departments

ii) Each department can offer any no. of courses

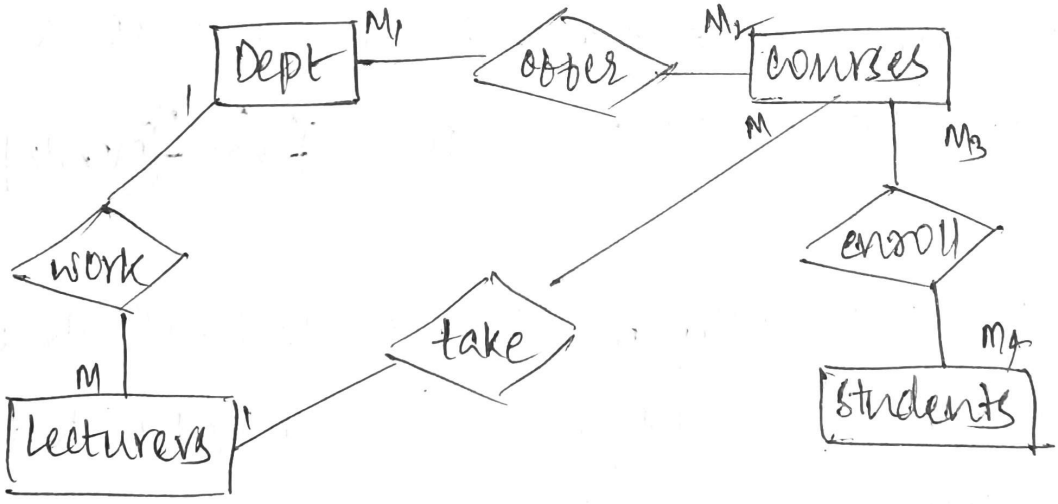
iii) Any no. of instructors can work in a dept.

iv) Instructor can only work in one dept.

v) Each instructor can take any no. of courses

vi) Students can be enrolled in any no. of courses

vii) Each course can have any no. of students.



a) A company has several departments.

Each dept has supervises and atleast one employee. Each employee must be mapped to atleast 1 but possibly more departments. Atleast 1 employee is assigned to 1 project but an employee may be on vacation.

11-6

* Integrity Constants Over Relations:

i) Key constant: It is a field or set of fields which uniquely identify a tuple or a row in table. Types of keys: i) Primary Key

a) Primary Key: This is a column / set of columns which can identify the rows of a table uniquely. The values that are placed in a primary key attribute should be unique, i.e; it does not support the concept of duplicates and also NULL values are not entertained while working with PK.

Syntax: CREATE TABLE tableName (col1 datatype, col2 datatype, primary key (col1));

Eg: CREATE TABLE employee (eid int(10), ename VARCHAR(10), eage (int(3), primary key (eid));

(OR) CREATE TABLE employee (eid int(10) primarykey, ename VARCHAR(10), ...);

Eg: student (sname, sage, sdept, sgender);

⇒ Add column so, (OR) take primary key (sage, sname)

Eg:

A	10
A	5
B	10
B	5

b) Foreign Key: This is a column which points to the primary key of another table and a foreign key provides a link between two tables. Syntax: CREATE TABLE tablename col1 datatype, col2 datatype, foreign key(col1), references table1 (primary key));

Eg: CREATE TABLE stud (sid int(10), sname VARCHAR(10), primary key (sid));

CREATE TABLE accounts (roll sid int(10), sfee int(10), date_paid DATE, foreign key (sid) references stud (sid));

stud

sid	sname
01	A
02	B
03	C

accounts

roll	sfee	date_paid
01	10K	5th apr
02	10K	6th apr
03	20K	9th apr
03		
03		
02		
01		

→ Write a query to display the names of students who have paid their fees.

194

* Some other constraints:

1) NOT NULL: This make sure that a column value does not hold a NULL value, i.e; when we do not provide a value for a particular column while inserting record into a table it takes a NULL value by default. In order to overcome this, we use a "NOT NULL" constraint using the following syntax:

Create Table tableName (col1 datatype NOT NULL);

Eg: Create Table employee (eid int(5) NOT NULL, ename VARCHAR(10), eage int(5) NOT NULL);

2) UNIQUE: Unique constraint enforces a column value or a set of column values to be unique, i.e; if a column has a unique constraint it means that the column cannot have duplicate values in it. Syntax:

Create Table TableName (col1 datatype, col2 datatype unique);

Eg: Create Table employee (eid int(5), ename VARCHAR(20) Unique);

3) DEFAULT: This constraint provides a default value to a column when there is no value provided to that particular column while inserting a record into a table. For example:

Eg: Create Table Employee (eid int(5) default 1000);

4) CHECK: This constraint is used for specifying range of values for a particular column of a table. When this constraint is being used with the column it ensures that the column must have the value specified in the condition.

Create Table TableName (col1 datatype, col2 datatype CHECK (col2>100));

Eg: Create Table Employee (eid int(5) NOT NULL, ename VARCHAR(10), eage int(2) NOT NULL CHECK (eage>20));

* DOMAIN CONSTRAINTS: They get violated only when a given value of the attribute does not appear in the corresponding domain OR

incase it is not of the appropriate datatype. The datatype of the domain includes string, character, integer, date, time, etc.

→ ENTITY INTEGRITY CONSTRAINT:

It states that the primary key value cannot be taken as a NULL value, this is because the primary key is used to recognise the individual rows of a table and if the primary key takes a NULL value then we cannot identify those rows.

20-4

* AGGREGATE FUNCTIONS IN MYSQL:

1) count(): It displays the no. of tuples that are present in given table based on query entered by the user. It takes the following form:

SELECT count(COL_name)/col(*) from TableName;

SELECT count(sid) FROM stud;(--4)

SELECT count(*) FROM stud;

sid	sname
01	10
02	20
03	15
04	20

2) Average(): It gives you the average value for the particular field name and it takes the following form: SELECT AVG(colName) from TableName;

SELECT AVG(sid) FROM std;

3) MAX(): It is used when we want to find the maximum value from a given set of values present in a column. It takes following form:

SELECT MAX(colname) from TableName;

4) min(): It is used to find the minimum value from a given set of values present in a column of table. It takes form: SELECT min(colname) from TN;

5) sum(): It adds up all the values of a given field and displays the result. It takes form:

SELECT SUM(colname) FROM TableName;

→ Write a query to display the maximum age of student from student table.

SELECT MAX(sage) from Std;

→ Write a query to display the ~~name~~ sid of student with second highest age.

* DATE FUNCTIONS: Display the current time of system

SELECT curtime();

Display current date : SELECT curdate();

To get particular name of the day for a date

SELECT dayname(date);

Display diff b/w 2 dates: SELECT datediff(d1, d2);

Extracting year from date:

* DATABASE DESIGN: It includes the following steps

Required Analysis

↓

Conceptual DB Design

↓

Logical DB Design

↓

Schema Refinement

↓

Physical DB Design

↓

Security Design

21/4

* View: A view is termed as virtual table which is based on the results set of an SQL statement. A view usually consists of rows and columns and the fields of a view are the fields from one/more tables.

create view, view_name as select col1, col2, ...
from tablename where condition;

→ create view view1 as select cid, cname, cadd from Cust;

SELECT * from view1;

Insert into view1 values (05, 'Amit', 'Hyd');

22/4

→ Create a view to insert the details of all the computers whose age is greater than 25.

Query: CREATE view v1 as SELECT * from cust WHERE cage>25;

→ Write a query to display the details of customers whose name starts with 'a'.

*Keywords in SQL:

1) alter: It is used to add, delete, modify a column

2) and: It only includes the rows where both conditions are true. Syntax: SELECT * FROM TableName WHERE condition1 AND condition2;

3) AS: Used to rename a column / a table with an alias name.

4) Between: Used to select values within a given range.

Syntax: SELECT * FROM TableName WHERE colname BETWEEN val1 and val2;

5) Create: It is used to create a database, an index, a view, a table, a procedure, etc.

6) DELETE: It is used to delete rows from a table based on the fineness of the condition.

7) Distinct: It is used to select only distinct values from column.

8) DESC: Describe

9) IN: It is used when you want to specify multiple values in a WHERE clause.

SELECT * from Cust WHERE Sage is {10, 20, 30};

10) IS NULL: It is used to test for empty values in a column.

SELECT * from CUST WHERE cname is NULL;

11) LIKE: It searches for a specific pattern in a column. You can work with it in two ways: '%' (which represents 0 or 1 or multiple characters) and '_' (underscore) (It represents single character). Mostly works with strings.

→ Write a query to display details of customers whose name starts with 'a'.

SELECT * from CUST WHERE cname like 'a%';

ending with a ⇒ cname like '%a';

→ Details of customers who have characters ab in any position of their name. ⇒ like '%ab%'

→ Write a query to display details of customers whose name starts with 'a' and is of length 3.

12) Limit: Specifies the no. of records that needs to be written as a result set. Syntax: SELECT * from Tname Limit 2;

13) NOT:

Ans:

→ ORDER BY: In order to sort rows in the result set we add the order by clause in the select statement using:

SELECT selectlist FROM TableName ORDER BY

ColName [ASC/DESC];

Ex: SELECT * FROM cust ORDER BY cname ASC;

(OR) SELECT * FROM cust ORDER BY cname;

→ Default sorting order is ascending (ASC).

Eg: SELECT * FROM cust ORDER BY cname Asc, cadd DESC;

* STRING FUNCTIONS IN MYSQL:

1) Concatenation: We use the keyword CONCAT which:

SELECT CONCAT (str1, str2);

2) Instr(): Sometimes you want to locate a substring in a string or you want to check if a substring exists in a string. In this case, we use the above function:

Instr (str, substr);

eg: SELECT INSTR('keshav memorial', 'keshav');

Output: 1

3) Length: This is used to get the length of a given string: SELECT Length (stringname);

eg: SELECT length('HELLO'); Output: 5

4) LOWER and UPPER: These functions are used whenever you want to convert a given string into lowercase/uppercase letters used for display purpose only. Syntax:

SELECT lower('Hello'); Output: hello

SELECT upper('world'); Output: WORLD

5) LEFT: is used to get a specific number of characters from the left part of the string.

SELECT left (string, length);

eg: SELECT left ('keshav', 3); Output: kes

SELECT left ('keshav', 0); Output: —

SELECT left('keshav', -3); Op: —

SELECT left('keshav', Null); Op:

6) Replace: Allows you to replace a string in a column of a table by the new string. It takes the form:

SELECT Replace('SQL Tutorial', 'SQL', 'HTML');

Output: HTML tutorial

7) Reverse: It is used to reverse a string and display the result.

→ Consider the following schema: customers with cid, cname, caddr, city, pin and country as its attributes. Insert 5 tuples into the table and write the following queries:

i) Display details whose country is India, city is Hyd.

ii) Display details of customers who live either in India or Hyderabad

iii) Details of customers who dont live in Germany.

iv) Details of customers whose country is Germany and city is Hyderabad / Berlin.

v) Write a query to display first five characters of the customer countries.

vi) Write a query to display details of cust table with

ascending order of id.

vii) Display combination of customer city & country.

viii) Display length of all the cities in the table.

ix) Write a query that fetches unique value of countries from customer table and print length.

x) Display city from cust table after replacing 'a' with 'A' if it exists.

xi) Display city from cust after removing white spaces from the right side.

xii) Write a query to display first 3 rows of cust.

xiii) Display customer details with DESC order of pin and ascending order of city.

xiv) Display total no. of customers who are from Hyderabad.

xv) Write a query to display details of second maximum pin.

Q7/4

→ Consider the following schema: Employee with (Eid, Ename, Bdate, Address, Sal, Deptid);

Dept (Depid, Depname, Depaddress);

Answer the following queries:

i) Display the total amount spent on the employees

ii) Display information of employees who works in D01.

iii) Show resulting salaries of each employee after every employee has been given a hike of 10%.

iv) Display information of employees whose Depid lies in D01, D02, D05.

Queries: i) SELECT SUM(Sal) FROM Employee;

ii) SELECT * FROM Emp, Dep WHERE Emp.Depid = Dep.Depid;

iii) SELECT Eid, Sal * 1.1 FROM Emp;

iv) SELECT * FROM Emp, Dep WHERE Dep.Depid in('D01', 'D02', 'D05');

→ v) Update address to Hyderabad whenever sal > 10000 along with depid 'D05'.

UPDATE Emp, Dep SET Emp.Addr = 'Hyderabad' WHERE Emp.Sal > 10000 AND Emp.DepId = 'D05';

→ vi) Write a query to remove details of dept with depid 'D01'.

DELETE FROM Dep WHERE DepId = 'D01';

(This gives an error because Depid is foreign key)

* conceptual Design Using ER model:

Concept of ER model is used in this stage and we have to analyze what are the different kinds of entities and relationships that are existing in an enterprise. What information about these entities and relationships needs to be stored into the database. Find out what are the integrity constraints and business rules that hold. We also need to test whether an ER model can be mapped into a relational model along with checking the schema for redundant data.

→ Design choices: Whether a concept should be modelled as an entity or an attribute, as an entity or a relationship. Try to identify the relationships (binary or ternary) and trying to analyse whether aggregation can be used or not. The problems with ER model would be some of the constraints cannot be captured in ER models.

→ Entity vs Attribute: Consider student as an entity. Attributes are sid, sname, Age, gender, address. Address can act as an entity too because a student can live at multiple addresses.

→ Draw an ER model for hospital management system:

28/4

→ consider the following schema (cid, name, address, city, pin code, country) customer and supplier (sid, sname, address, city, pin code, country). Write a query that returns the cities (distinct only) from both customers and suppliers' table.

29/4

* ON DELETE CASCADE: It is used to automatically remove the matching rows from the child table. Whenever we delete corresponding rows from the parent table.

CREATE TABLE Table1 (A1, A2, A3, PK(A1));

CREATE TABLE Table2 (A1, A2, A3, A4, FK(A) referencing PK(A1 wrt Table1) ON DELETE CASCADE);

(UPDATE)

(Update in child table if data is updated in parent table)

* Used to update matching data from child table whenever we are trying to update rows in parent table.

Syntax: CREATE TABLE table2 (A1, A2, A3, A4, FK(A)

Referencing PK(A1 wrt Table1) on update cascade;

* UNION: It is an operator that allows us to combine the results from multiple select statements into a single result set. By default, it removes all the duplicate values from the result and it usually uses the name of the columns from the first select statement. It takes the following form:

SELECT COL_list FROM Table1 WHERE condition UNION
SELECT COL_list FROM Table2 WHERE condition;

To show even duplicates: UNION ALL

* Intersect:

SELECT COL_list FROM Table1 WHERE CON INTERSECT
SELECT COL_list FROM Table2 WHERE condition;

→ Write a query to perform intersect of 2 tables wrt id in mysql.

```
SELECT distinct(id) FROM table1 WHERE id in
    (SELECT id FROM Table2);
```

→ Write a query to display the customer id's of customers and suppliers in ascending order.

* GROUP BY: The groupby statement groups the rows that have same value into summary rows like finding the no. of customers in each country or finding the no. of sailors for a particular rating. 'Group By' is often used with the aggregate functions in order to group the result set by one or more columns. It takes the following form:

```
SELECT col_list FROM TableName WHERE Condition
    GROUP BY (col_name);
```

21/5

→ Having clause: It is used in combination with group by clause and it returns the rows whenever condition is true. It takes the form:

select selectlist Aggregate functions

FROM tableName [where cond]

Group by colname having conditions;

Q) Consider the schema customers with cust (cid, cname, caddr, city, pin, country); write a query to list the number of customers in each country and include countries with a minimum of 5 customers in it.

Query: select count(cid) from cust group by country having count(cid) > 5;

Same result in descending order:

select count(cid) from cust group by country having count(cid) > 5 order by count(cid) DESC;

→ NESTED QUERIES: (Sub Queries)

A subquery is a query which is nested inside another query and embedded with select, update or delete statements. Points to remember:

i) A subquery is always used in parenthesis

ii) subquery is valued first and the result is used by the main query.

iii) The main query is termed as the outer query and the sub query is termed as the inert query.

iv) We can use various comparison operators such as greater, equal to, less than, any, sum, all, in, not in, etc.

v) >all, >any, <all, <any are some combinations

Syntax for working with nested queries:

```
select collist from tablename where colname operator
(select colname from tablename where condition);
```

→ Write a query to display the employee details with max sal.

```
select * from employee where income = (select max(income)
from employee);
```

→ Write a query to display the second highest income of an emp

```
select * from emp where income not in (select max(income)
from emp);
```

* ANY, ALL IN MYSQL:

ANY and ALL operators allows you to perform a comparison bw a single column value and a range of other values.

Any operator: It returns true if any of the subquery results meets the condition; i.e; the condition will be true if the operation is true for any one of the values in the range. Syntax:

Select colname from tableName where colname operator ANY/ALL (Subquery);

→ Write a query to display the details of sailors who did not reserve a red boat.

Select cid from customer whose cid not in (select cid from reservation);

Select cid from customese where cid in (select cid from reservations);

QUESTIONS:

1) Write a query to display the colours of boats reserved by Dustin.

2) Write a query to display the name and age of the youngest sailor using subqueries.

3) Find the average of sailors for each rating level.

5/5

* Southwind Database: Sample database used by organisation. The database contains sales data for Southwind traders. It is a food export import company. Using this schema demonstrate how a customer can choose and order products, how orders are placed and how those products get delivered to the customers.

1-b

* Schema Refinement: It is process of refining the schema so as to solve the problems caused by redundantly storing the information into the database. This can be done by decomposing larger schema into smaller ones.

Redundancy: It occurs when there is a copy or rewriting of information, i.e; when same data is stored into the database again and again. It leads to unnecessary data being entered and may lead to the following problems:

i) Wastage of memory space (Increases DB size)

ii) Possibility of inconsistent data

iii) It might lead to more difficult data updates

iv) Data access becomes slower

v) Leads to certain anamoly which are problems that occur in unnormalised databases wherein all the data is stored into a single table.

Types of anamolies:

i) Insertion: Suppose you want to add new information into a table but cannot enter the data due to certain constraints, then it leads to the concept of insertion anamoly.

Stud (Sid, Sname, Saddr, Did, Dname, Clg, HOD).

01	A	Hyd	01	CSE	KMIT	X
02	B	Hyd	01	CSE	KMIT	X
03	C	Hyd	02	IT	KMIT	Y
04	D	Hyd	02	IT	KMIT	Y
?	?	?	03	DS	KMIT	Z

2) Deletion: Occurs when certain attributes info is lost while there is a requirement of deletion of some other attributes in the table. Eg: Deletion of 03 (Sid) in above table

3) Updation: Occurs when the change of value in a single attribute may require changes in multiple records otherwise may lead to concept of inconsistent data in the database.

01	A	Hyd	01	CSE	KMIT (X) Y
02	B	Hyd	01	CSE	KMIT (X) Y
03	C	Hyd	01	CSE	KMIT (X) Y

Update HOD name from 'X' to 'Y'.

* Normalisation: It is a step by step process through which we can decompose/divide a table into more than one relation in order to work with concept of anamalies and reduce redundancy to make database more efficient.

We make use of following normal forms:

1) First NF
2) Second NF
3) Third NF
4) 3.5 or BCNF (Boyce Codd Normal Form)
5) Fourth NF
6) Fifth NF

Q-6

* Functional Dependency:

Consider a relation R with X and Y as subset of the set of attributes of R. Then we say that Y is functionally dependent on X if a given value of X uniquely identifies the value of Y. It is represented by $X \rightarrow Y$. Eg:

A	B
a_1	b_1
a_2	b_2
a_3	b_3
a_1	b_1

$A \rightarrow B$ (✓)

Types of FD | Classification of FD:

1) Trivial FD: A FD $X \rightarrow Y$ is trivial if Y is a subset of X.

For eg: R(ABC), $AB \rightarrow A$ or B; Sid, Sname $\rightarrow$ Sid

2) Non Trivial FD: If there is atleast one attribute in the right hand side of the FD which is not a part of LHS then we say it is non-trivial FD. For eg:

Stud(Sid, Sname, Sage)

Sid, Sname $\rightarrow$ Sage, Sid $\rightarrow$ Sage, Sid $\rightarrow$ Sname

3) Fully Functional: Given Relation R and FD $X \rightarrow Y$ then we say Y is fully functional dependent on X if there exists no Y which is dependent on a proper subset of X. Eg:

Sid, Sname $\rightarrow$ Sage (X)

Sid $\rightarrow$ Sage (✓)

4) Partial FD: Given relation R and FD $X \rightarrow Y$ then Y is partially FD on X if there is a Z which is a proper subset of X such that Z determines Y.

Eg: Sid, Sname $\rightarrow$ Sage (✓)

5) Transitive dependency: This occurs when we have atleast 3 columns in our table. For eg:

$A \to B$ and $B \to C \Rightarrow A \to C$.

* Armstrong Axioms: These are the basic set of inference rules used for finding out new, logically implied functional dependencies of relation.

i) Reflexive Rule: If x is a subset of y then y implies x.

$x \subseteq y \Rightarrow y \to x$. It means $x \to x$.

ii) Augmentation / Addition Rule: If $x \to y$ then $xz \to yz$

iii) Transitive Rule: $x \to z$ (new FD)

$x \to y$ and $y \to z$ (existing)

iv) Union Rule: If $x \to y$ and $x \to z$, then $x \to yz$.

v) Productive Rule / Decomposition:

If $x \to yz$, then $x \to y$ and $x \to z$.

* Attribute closure: The set of all attributes that can be determined is called as Attribute closure. It is represented by $(attribute)^+$

Eg: R(ABC), A → B and B → C

$A^+ \Rightarrow ABC$, $B^+ \Rightarrow BC$, $C^+ \Rightarrow C$.

6-b

* 1NF: The relation R is said to be in 1NF, if the domain of each attribute contains only atomic values.

Example for a table which is not 1NF is:

(Student)

Sid	Sname	Smarks
01	A1	M1, M2, M3
02	A2	M1, M2
03	A3	M1

→ (x)

(1)

(Student)

Sid	Sname	Smarks
01	A1	M1
01	A1	M2
01	A1	M3
02	A2	M1
02	A2	M2
03	A3	M1

(std1)

(2)

Sid	Sname
01	A1
02	A2
03	A3

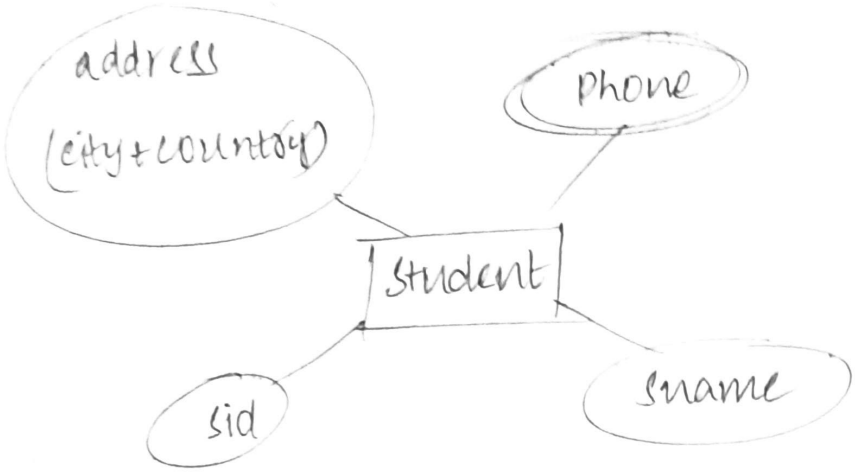
Sid	Smarks
01	M1
01	M2
01	M3
02	M1
02	M2
03	M1

(std2)

(3)

Sid	Sname	SM1	SM2	SM3
01	A1	M1	M2	M3
02	A2	M1	M2	-
03	A3	M1	-	-

Eg:



Student (sid, sname, city, country P_1 P_2 P_3 ....);

* 2NF: A relation R is in 2NF, if it is in 1NF and there is no concept of partial dependency.

Eg: Consider a relation R with A, B, C, D, E, F as its attributes with the set of functional dependencies:

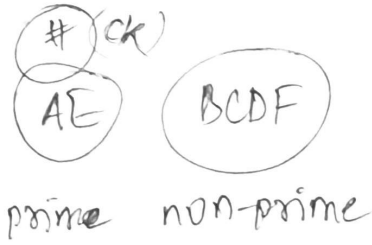
$A \to B$, $B \to C$, $C \to D$, $D \to F$.

Find whether R is 2NF or not assuming R is in 1NF.

A) $(ABCDEF)^+ = ABCDEF$ (SK) [A B C D E F $\Rightarrow$ AE]

$(AE)^+ = AEBCDF$ (SK and CK)

$A^+ = ABCDF$ and $E^+ = E$



$\Rightarrow$ This is not in 2NF

since $A \to B$ [not $AE \to B$].

$\to$ [B is partially dependent on AE].

Eg2: R(ABCD), AB → CD, C → A, D → B.

$(ABCD)^+ = ABCD$ — (SK) [AB ∉ ⊕ ⇒ AB]

$(AB)^+ = ABCD$ — (SK), (CK)

A^+ — A

B^+ — B

→ In 2NF.

PA	NPA
AB	CD

$(CB)^+$: CBA, CBAD

$(AD)^+$: ADB, ABCD

Eg3: Relation R(ABCD) with A → B, B → C, C → D.

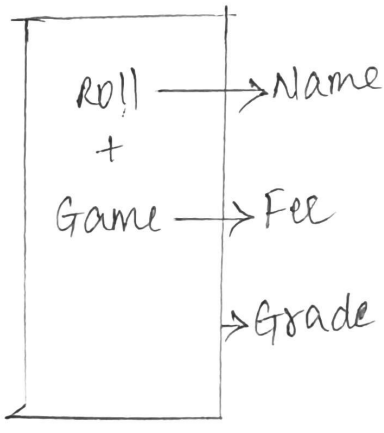
$(ABCD)^+ = ABCD$ — (SK) [A ∉ ⊕ ⇒ A]

$(A)^+ = ABCD$ — (SK), (CK).

PA	NPA
A	BCD

A^+ : ABCD

Eg4: R(Roll, Name, Game, Grade, Fee);



{Roll → name
game → fee
(roll + game) → grade}

Roll	Game	Name	Fee	Grade
1	Cricket	Amit	200	A
2	Badminton	Deeraj	150	B
3	Cricket	Lalit	200	A
4	Badminton	Prafull	150	C
5	Hockey	Jack	100	A
6	Cricket	John	200	C

(1)

ROll	Name
1	Amit
2	Deeraj
3	Lalit
4	Praful
5	Jack
6	John

(2)

Game	Fee
Cricket	200
Badminton	150
Hockey	100

(3)

ROll	Game	Grade
1	cricket	A
2	Badminton	B
3	Cricket	A
4	Badminton	C
5	Hockey	A
6	Cricket	C

8-6

* 3NF (Third Normal Form): A relation R is said to be in 3NF if the relation is in 2NF and there is no concept of transitive dependency. (OR) A table is in 3NF if and only if for each of its non-trivial functional dependencies atleast one of the following conditions hold: i) LHS is a super key
ii) RHS is a prime attribute

Eg: R(ABCD), A→B, B→C, C→D ⇒ 3NF?

(PA) (NPA)
A BCD

$(ABCD)^+ = ABCD$ (SK) [A→BCD] ⇒ A

$(A)^+ = ABCD$ (SK) (CK) {It is in 2NF}

M1: Here, if NPA→NPA, then not in 3NF {B→C(X)}

M2: LHS SK, RHS PA ⇒ B→C and C→D are violation

⇒ NOT IN 3NF.

Eg: Find whether given relation is in 3NF, if not decompose the table to 3NF:

(Item + Color is CK)

Item	Color	Price	Tax (R)
T-shirt	Red, Blue	12	.60
Polo	Red, Yellow	12	.60
Swet-shirt	Blue, Black	25	1.25

1NF

Item	Color	Price	Tax
Tshirt	Red	12	.60
Tshirt	Blue	12	.60
Polo	Red	12	.60
Polo	Yellow	12	.60
Swetshirt	Blue	25	1.25
Swetshirt	Black	25	1.25

(NPA) (NPA)

Item Price Tax (R2) — 3NF

Item Color (R3) — 2NF

Decomposition:

Item, Price

Price, Tax

* Steps to find whether a decomposition is lossy/lossless:

Step 1: There are atleast two rows with distinguished variables for the LHS of a functional dependency.

Step 2: There is atleast one row with a distinguished variable for the RHS of a given FD.

Step 3: There is atleast one row with a non distinguished variable for the RHS of a functional dependency.

If all the above steps are satisfied then you can add a distinguished variable.

9b

Eg: R(ABCDE), AB → C, C → E, B → D, E → A

Decomposed into R1(BCD), R2(ACE)

	A	B	C	D	E
R1	$\bar{a}$	a	a	a	$\bar{a}$
R2	a		a		a

AB → C step1 ✗

C → E step1 ✓ step2 ✓ step3 ✓ ⇒ $\bar{a}$

B → D step1 ✗

E → A step1 ✓ ⇒ $\bar{a}$

R1 is filled ⇒ Lossless

Eg: R1(ACD), R2(DC), A → B, B → C, C → D

	A	B	C	D
R1	a		a	a
R2		a	a	

A → B ⇒ ✗

B → C ⇒ ✗ (LOSSY)

C → D ⇒ ✓

Eg: R(ABCDE), R_1(ABCD), R_2(CE),

AB → CD, D → A, C → E

	A	B	C	D	E
R_1	a	a	a	a	$\overline{a}$
R_2			a		a

AB → CD ⇒ ✗ (Lossless)

D → A ⇒ ✗

C → E ⇒ ✓, ✓, ✓

* Boyce Codd Normal Form (BCNF): A relation R is said to be in BCNF if the following conditions hold:

i) R is already in 3NF

ii) For each non-trivial FD, X → Y; X must be a superkey.

1Eg: consider a relation R(ABC), A → B and B → C and C → A.

Find whether R is in BCNF, assuming it to be in 1NF.

A) A^+ = ABC, B^+ = BCA, C^+ = CAB (All are CK) ⇒ (SK).

A, B, C are Super Keys ⇒ In BCNF.

2Eg: Find the highest normal form if given relation is in wrt set of FDs given R(ABCDE), A → BCDE, BC → ACE, D → E

A) $(ABCDE)^+$ = ABCDE — (SK) [ABCDE]

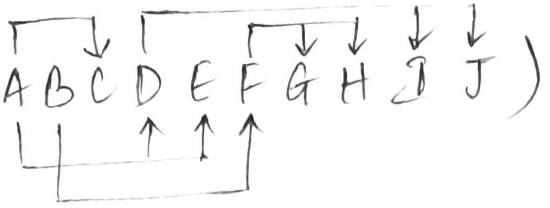
PA: (A) NPA: (BCDE)

A^+ = ABCDE — (SK, CK) (2NF)

NPA → NPA ⇒ D → E (3NF) ✗

⇒ Not in BCNF

Eg: R(A B C D E F G H I J)



$A \rightarrow CDE$, $F \rightarrow GH$

$B \rightarrow F$

$D \rightarrow IJ$

$A^+ = ACDEIJ$

$B^+ = BFGH$

$D^+ = DIJGH$

$F^+ = FGH$

10-b

i) R(ABCDEF), $AB \rightarrow C$, $C \rightarrow DE$, $E \rightarrow F$, $D \rightarrow A$, $C \rightarrow B$

Find the highest NF. [ABCDEF]

$(ABCDE)^+ = ABCDEF$ (SK)

2NF

PA: AB

NPA: CDEF

A^+: DB

B^+: AC

* 4NF: Relation R is said to be in 4NF if following conditions hold: i) should already be in BCNF

ii) there is no concept of multivalued dependency

-> Multivalued Dependency: It occurs when more than one independent multivalued attribute is present in the table. For a table to have MVD, min. no of columns = 3.

[$X \rightarrow\!\!\rightarrow Y$]

Eg: R(ABC), A → B, B → C

Assuming this is already in BCNF. NOT MVD.

If A → B and A → C ⇒ R1(AB), R2(AC) ⇒ (4NF) ✓.

Eg2: Sid → subject, Sid → Activity

Sid	Subject (MVD)	Activity (MVD)
100	music	Swim
100	ACC	Swim
100	Music	Tennis
100	ACC	Tennis
150	math	Jog

(4NF) ✗ MVD in sub and Activity

Sub and Activity are independent.

13)b

* 5NF: Relation R is said to be in 5NF if it is already in 4NF and it cannot be further non-loss decomposed. OR

5NF is satisfied when all the tables are broken into as many small tables as possible in order to avoid redundancy and once the relation R is in 5NF it cannot be further broken into smaller pieces/relations. Sometimes it is also referred as PJNF (Project Join Normal form). Eg:

{ R1(C1 C2), R2(C1 C3), R3(C2 C3) } ⟶

R

C1	C2	C3
C01	C22	C32
C02	C21	C31
C01	C21	C33

* TRIGGERS: A trigger is a set of SQL statements which is executed or fired whenever an event associated with a particular table occurs. It is used as an alternative to check the integrity of data and it takes the following form:

CREATE TRIGGER TRIGGER_NAME TRIGGER_TIME { BEFORE / AFTER }
TRIGGER_EVENT { INSERT / DELETE / UPDATE } ON TABLE_NAME
FOR EACH ROW BEGIN
(delimiter #)
= } LOGIC OF TRIGGER
END

15/6

Q) Student (Sid, Sname, Saddress, marks). Create a trigger that will add 100 marks to each of the row that is being inserted into marks column whenever a new student is inserted into the table.

2Q) Create a table called salary with enumber and esal as its attributes. Enter 5 records into the table. create one more table called salary budget with total as its attribute. the value of total being sum of salaries from salary table. Then, create a trigger on salary budget such that if any deletion of records is done, then it should be reflected in the total column.

3) Create a table called student with name and age as its attributes. create a triggers to check value of age being entered if age is entered a negative value, then it should be inserted as 0.

4) Create a trigger to duplicate values inserted into one table that should be reported in another table.