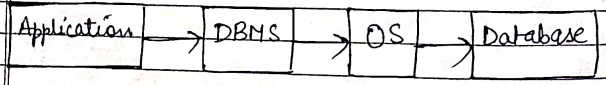
DBMS

Date Page 15th April

Introduction of Database :-

1) It is the collection of inter-related data and a set of program to access those data.
2) It is used to organise the data in the form of table, schema, view and report
3) It is a primary goal to provide a way to store and retrieve database information i.e. both convinient and efficient.
4) Database management is the combination of two words. first is Database and second is Management.
5) It is the collection of related information stored, so that it is available to many users for different purpose.
6) It is also a collection of program that enables users to create and maintain the database.



* This interface b/w the application program and OS to access and manipulate that database.
* DBMS is a software which is used to manage database. for ex- MySQL, ORACLE etc. are very popular commercial databases which is used in different application.

Date Page

Characteristics of DBMS :-

1) It uses the digital repository established on a server to store and manage the information.
2) It can provide a clear and logical view of the process that manipulate data.
3) It contains automatic backup and recovery procedure.
4) It contains ACID properties which maintain data in a healthy state in case of failure

A → Atomicity
C → Consistency
I → Isolation
D → Durability

5) It can reduce the complex relationship b/w data.
6) It is used to support manipulations and processing the data.
7) It provides security of data.

Applications of DBMS :-

1) Banking :- for maintaining customer's information, A/c, loans and banking transactions.
2) University :- It maintains student's records, courses and grades.

3) Railway :- It checks availability of reservations in different trains.

4) Airlines :- For reservation and schedule information.

5) Telecommunication :- It keeps record of call made, generates monthly bills etc.

6) Finance :- It stores information about holidays, sales and purchase of financial instruments.

7) Sales :- For customer, product and purchase information.

Advantages of DBMS :-

1) Control database redundancy → It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.

* 2) Data Sharing → In DBMS the authorized users of an organization can share the data among multiple users.

3) Easy Maintainance → It can be easily maintainable due to the centralized nature of the database system.

4) Reduce Time → It reduces development time and maintain once need.

* 5) Back up → It provides backup and recovery subsystems which create automatic backup of data from h/w and s/w failure and restores the data if required.

* 6) Multiple User Interface → It provides different type of interface like GUI, API.

Disadvantages of DBMS :-

1) Cost of h/w and s/w
2) Size
3) Complexity
4) Higher impact of failure

Types of Database :-

1) Centralized → a) It is a type of database that stores the data.
b) It comfort the user to access the stored data from different locations through several applications.
c) The appl" contains the authentication process to let users access the data security.

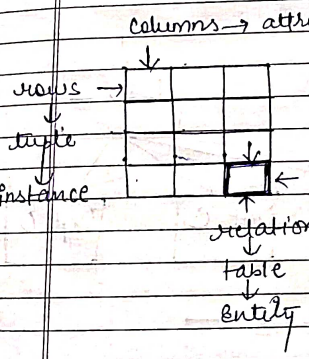
2.) Distributed → a) In this database, data is distributed among different database system of organization.
b) These database systems are connected via communication link such as link helps to end user access data easily.
c) It has two type → i) Homogeneous
ii) Heterogeneous

3.) NoSQL → a) Not Only SQL
b) It is a type of database that is used for storing wide range of dataset.
c) It is not a relational database as it stores data not only in tabular form but in several different ways.
d) It also divided into four parts →
i) Key values storage
ii) document oriented database
iii) graph database
iv) Wide column store

4.) Cloud Database → a) It is a type of database, where data is stored in a virtual environment and execute over the cloud computing plateform.
b) It provides the user with various cloud computing services like SAAS, PAAS, IAAS etc for accessing the database.
c) Other example of database such as AWS, MS Azure, Google cloud SQL.

5.) Relational DBMS (RDBMS) →

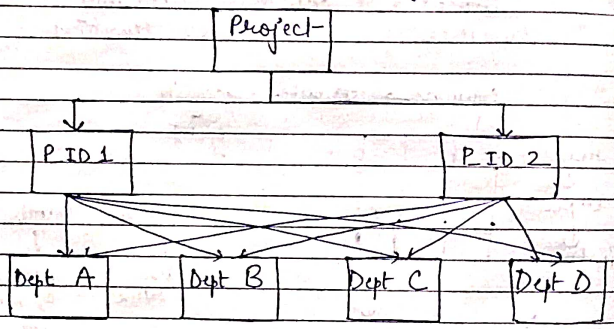
a) It stores the data in the form of rows (tuple) and columns (attribute) and together form a table. (domain).



Relational → form table → I
Database → collection of data → II
Mgmt System → ways to manage data → III

b) The relational database uses SQL for storing, manipulating as well as maintaining the data.
c) Each table in the database carries a key that make the data unique from others.
d) MySQL, MS-SQL server, Oracle etc.

6) N/w Database → a) This database typically follows the n/w data model.
b) In this representation of data is in the form of node connected by links b/w them.
c) It allows each record to have multiple children and parent node to form a generalized graph structure.



7) Object Oriented Database →

a) It uses the object based data model approach for storing data in the database system.
b) It is represented and stored as object which are similar to object used in the Object Oriented Programming Language.

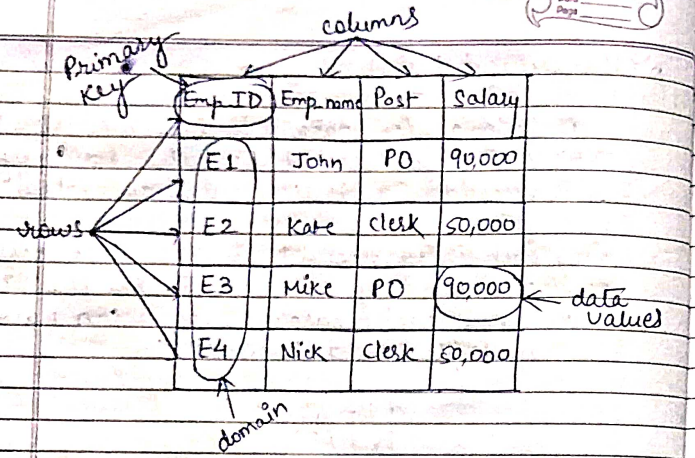
8) Hierarchical Database →

a) It organise data in tree like structure.
* b) Data get stored in the form of records and that are connected via links.
c) Each child record in the tree will contain only one parent. On the other hand, each parent record can have multiple child records.

RDBMS

1) Introduced by Dr. E.F. Codd.
2) It stores the data in the form of related tables.
*3) It is an important feature of relational systems is that a single database can be spread across several tables.
4) All modern DBMS
   - MySQL
   - MS-SQL Server
   - IBM DB2
   - ORACLE

   } Microsoft access
5) In relational database most commonly used database. It contains several tables and each table has it's primary key.
6) Everything in the relational database is stored in the form of relations.



Properties of Relational Table :—

1) Values are atomic.
2) Column values are of the same kind.
3) Each row is unique.
4) Each column has a unique name.
5) The sequence of the rows and columns is insignificance.

Properties of Rows / Records :—

1) No two tuples are identical to each other in all their entries.
2) All tuples of relation have same format and the same no. of entries.
3) The order of tuple is irrelevant, they are identified by their content not by their position.

Properties of attribute :—

1) Every attribute of a relation must have a name.
2) Null values are permitted for attribute.
3) Attribute that uniquely identify each tuple of a relation are the primary key.
4) Default value can be specified for an attribute automatically inserted if no other value is specified for an attribute.

Data Icons / Cells :—

1) The smallest unit of the data in the table is the individual data items.
2) It is stored at the intersections of tuples and attributes.

Degree :— Total number of columns.

Cardinality :— i) Total no of tuples
ii) The relation whose cardinality is zero, is called empty table.

Domain :— i) It refers to the possible values each attribute can contains.
ii) It can be specified using standard data type such as int, float etc.

Data Model - 1970
Launch - 1985

Date 16th April

12 Rules

1) Information Rule → This rule simply requires that all data should be presented in data form. This is the basis of relational model.

2) Guaranteed Access Rule → Every single data element (values) is guarranteed to be accessible logically with a combination of table name, primary key (rows values) and attributes (col^n name).

3) Systematic Treatment of Null Values →

The null values in a database must be given a systematic and uniform treatment. A null values can be interpretated.

* Data missing.
* Data is not known or data is not applicable.

4) Active Online Catalogue → The structure description of the entire database must be stored in an online catalogue. It is also known as Data dictionary which is access by authorized user. User can use the same query language to access the catalogue which they use to access the database itself.

5) Comprehensive Data Sub-Languages Rule →

A database can only be access using a language having linear syntax that support DDL (Data Definition Language), DML (Data Manipulation Language) and Transaction Management Operations.

6) View Updating Rule → Data can be presented in different logical combination called views.

* Full range of data manipulation
* Direct access to a table available.

7) High Level Insert, Delete and Update →

The database must support insertion, updation and deletion.

* It must not be limited to a single row.
* It must also support union, intersection and minus operations to yield sets of data records.

8) Physical Data Independance → The data stored in a database must be independent of the applications that access the database.

* Any change in physical structure of database must not have any impact on how the data has been accessed by external applications

9) Logical Data Independence → The logical data in the database must be independent of it user view (applications). Any changes in logical data must not effect application uses it.

10) Integrity Independence → The database language (SQL) should support constraints on user input that maintain database integrity.
* No component of a primary key can have null values.
* If foriegn key is defined in one table, any value in it must exist as a primary key in another table.

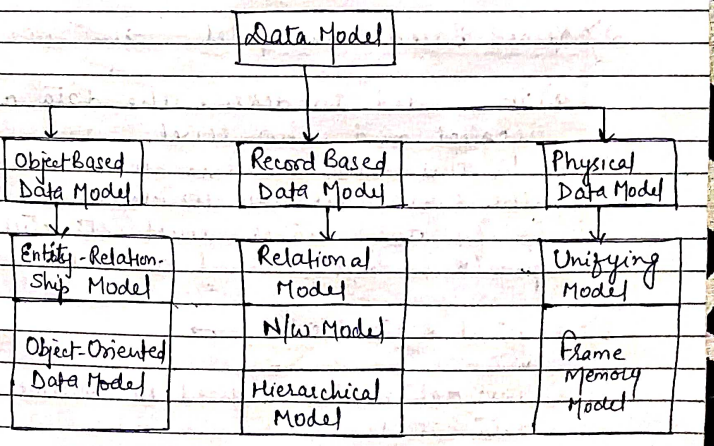
11) Distribution Independence → The end user must not be able to see the data is distributed over various locations. User should always get the impression that the data is located at. This rule has been regarded as a foundation of distributed database system.

12) Non-Subversion Rule → There should be no way to modify a database structure other than through the multiple row database languages (SQL). Most databases support by administrative rules that allow some direct manipulations of data structure.

Data Model :— Data model is the modelling of the data description, data semantic and consistency constraints of the data. Data model provides the conceptual tools for describing the design of a database at each level of data abstractions.
* It is defined a collection of high level data descriptions, construct that hide many low level storage details.

Types of Data Model :—

1) Object Based Data Model →
a) That model provides flexible structuring capability
b) It is used to describe the data at the logical and view level.



Entity Relationship Model → It is logical representation of data as object and relationship among them.
These objects are also known as entities and relationship is an association among these entities.

Object-Oriented Data Model → In this model, informations or data is displayed as an object and these objects store the values in the instance variable.
This model works with object-oriented programming language like Python, Java, Perl and C#.net.

2) Record Based Data Model →

a) It is used to describe the data at logical and view level.
b) In this data model is used to specify the overall logical structure and to specify the higher level structure and provided higher level description.

3) Physical Data Model →

a) This data model is used to describe the data at low level

Data Language (DBMS Language) :—

DBMS has four types of languages :-

1) DDL :— i) Data Definition Language
ii) Below tasks performed under DDL -

create → Used to create object in the database.

alter → used to alter the structure of database

drop → used to delete object from the database.

truncate → used to remove all records from a table including all spaces. Allocated for the records are removed.

comment → used to add comment to the data dictionary.

rename → used to rename an object

2) DML :- i) Data Manipulation Language
ii) Below tasks performed under DML-

select → It retrieves data from a database.

insert → It insert a data into a table.

update → It updates existing data within a table,

merge → It perform UPSERT operation such as insert and update operation

CALL → It is used to call a structure query language (SQL) or a java sub-program.

locktable → It controls concurrency.

delete → It deletes all record from a table.

3.) DCL :- i) Data Control Language
ii) Below tasks performed under DCL -

System → It creates session, tables are all types of system privilage.

Object → any command or query to work on table comes under object privilage.

grant → It gives user access privilage to a database.

revoke → It take back permission from the user.

4.) TCL :- i) Transaction Control Language
ii) Below tasks performed under TCL -

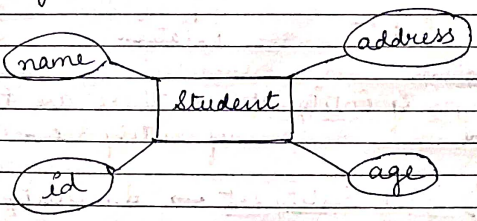
commit → It is used to save the transactions on the database.

Rollback → It is used to restore the database to original since the last commit.

ER-Model

It is based on perception of real world that consist of a collection of basic objects called entities and of relationship among these objects.

Eg.



Components of ER-Model :-

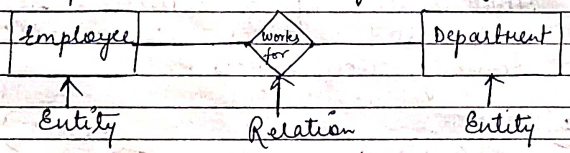
1.) Entity
- Weak Entity
- Strong Entity

2.) Attributes
- Key attributes
- Composite "
- Multivalued "
- Derived "

3.) Relation
- One to One
- One to many
- Many to one
- Many to many

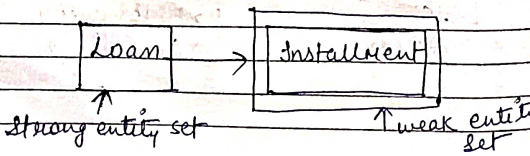
Entity → It is a thing or object in the real world i.e. distinguishable from all other object.
Anything about which we store information is called entity.

Entity Set → It is a set of entities of the same type that share the same properties and attributes.
* Entity is represented by rectangle.



Types of Entity Set → There are two types of entity set -

i) Weak Entity set → An entity that depends on another entity called a weak entity set.
* The weak entity set does not contain any key attributes of its own.
Weak Entity is represented by double rectangle.

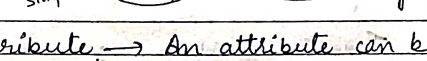


ii) Strong Entity Set → A strong entity set contain sufficient attributes to uniquely identify all its entities.
* Primary key exist for a strong entity set. Strong entity set is represented by single rectangle.

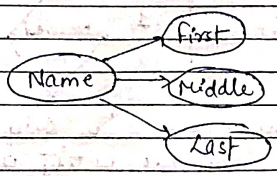
Attributes → It describes the property of entity.
In entity set may contains any no. of attributes.
It is represented by ellipse.

Types of Attributes →

i) Simple (Key) Attribute → An attribute that cannot be further sub-divided into components is called simple attribute.



ii) Composite Attribute → An attribute can be split into components is called composite attribute.



iii) Multivalued Attribute → It can have more than one values. These attributes called multivalued attribute. Double ellipse is used to represent multi-valued attribute.

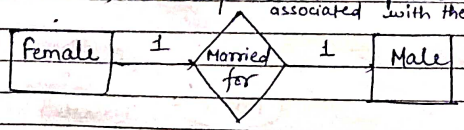
iv) Derived Attribute → It can be derived from other attributes so it is known as derived attribute. It represented by dashed ellipse.

Relation → It is used to describe the relation b/w two entities. It is also called mapping. It is represented by diamond or rhombus.

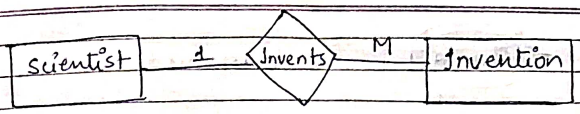
* Mapping cardinalities or cardinality ratio express the no. of entities to which another entity can be associated by a relationship set.

Types of Relation →

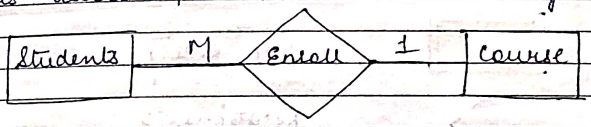
i) One to One → When only one instance of an entity is associated with the relationship.



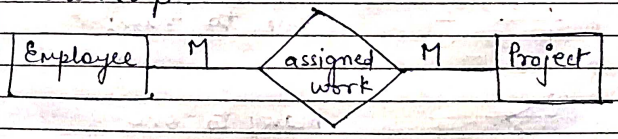
ii) One to many → When only one instance of the entity is on the left and more than one instance of the entity on the right is associated with the relationship.



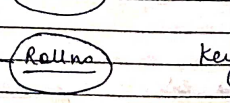
iii) Many to One → When more than one instance of the entity on the left and only one instance of an entity on the right is associated with the relationship.



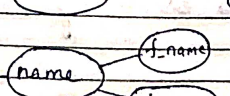
iv) Many to Many → When more than one instance of the entity on the left and more than one instance of the entity on the right is associated with the relationship.



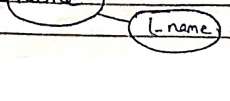
Notation of ER-Model :-



Attributes

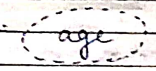


Key Attribute

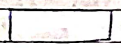


Composite Attribute

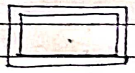
Multivalued



Derived



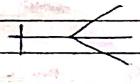
Strong entity



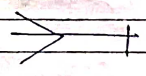
Weak entity

Relationship

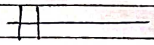
Links one to one



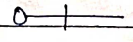
one to many



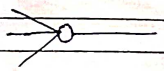
many to one



one to only one



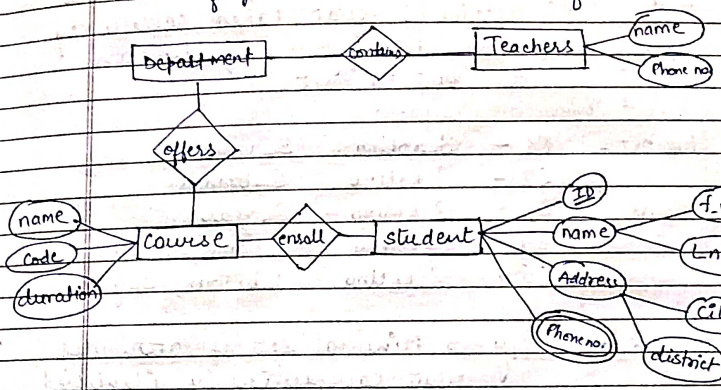
Zero or one

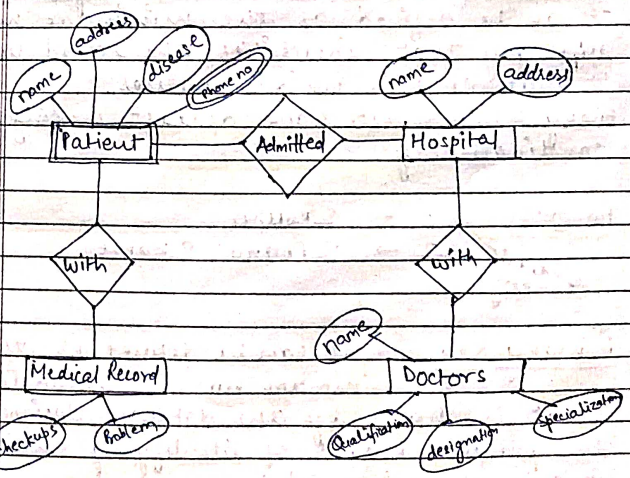


Zero or many

Q.1) Create an ER diagram of your college.

Q.2) Construct an ER diagram for a hospital with a set of patients and a set of medical doctors.





Keys of DBMS :—

1) **Super Key** → It is a set of one or more attribute that taken collectively allow us to identify uniquely an entity in an entity set.

For ex:- (Super Key)
- S1 - S_Rollno - S_name
- S2 - S_Rollno - S_branch
- S3 - S_Rollno - S_year
- S4 - S_Rollno - S_name, S_branch
- S5 - S_Rollno - S_branch, S_year

2) **Candidate Key** → Minimal set of attributes that can uniquely identified a tuple is known as candidate key.
It can be defined as the minimum no. of super key that identifies the record uniquely.
* It must contain unique values.
Every table must have atleast a single candidate key.

For ex:- (Candidate key)
- C1 → S_Rollno.
- C2 → S_Rollno, S_name

3) **Primary Key** → It can be defined as the minimum no. of candidate key that is chosen by the database designer as the principle means of identifying entities within an entity set.

* It is an unique key.
It can identify only one tuple at a time.
It cannot be null.
It can has no duplicate value. It has unique values only.
MUST * Primary keys are not necessary to be a single column, more than one column can also be a primary key for a table.

4) **Composite Key** → Whenever a primary key consist of more than one attributes is called composite key.

5) **Foreign Key** → It is a column whose values are the same as the primary key of another table.
It combines two or more relations at a time.
They act as a cross-reference b/w the tables.
They are the columns of the table used to point the primary key of another table.

Date 17 April
Page

Normalization

Functional Dependency :-

The functional dependencies is a relationship b/w two attributes.
It typically exists b/w the primary key and non-key attribute within a table.

determinant → A → B ← Dependent

The left side of dependency is known as determinant and the right side of the productions is known as dependent.

Types of Functional Dependency :-

1) Trivial F.D. → A → B has a trival dependency if B is the sub-set of A.
The following dependencies are also trival -
A – A
B – B
C – C

2) Non-Trivial F.D. → A → B has non trival dependency if B is not a sub-set of A.
When A intersection of B is null, A → B is called non-trival functional dependency.

Date
Page

Inference Rule (IR) :- It is a type of assertion. It can apply a set of F.D. to define other F.D.

1) Reflexive Rule → If Y is the subset of X, then X determined Y.

$Y \subseteq X \Rightarrow X \to Y$
↑ Reflexive Rule

2) Augmented Rule → In this rule, partially is called dependency.
If X is determinant Y, then XZ determines YZ.

$X \to Y$
$XZ \to YZ$

3) Transitive Rule → In this rule, if X determines Y and Y determines Z, then X must also determines Z.

$X \to Y$
$Y \to Z$
$X \to Z$

4) Union Rule → If X determines Y and X determines Z, then X must also determines Y and Z.

$X \to Y$
$X \to Z$.
$X \to YZ$

5) Decomposition Rule → This rule is also known as project rule. It is the reverse of union rule. If X determines Y and Z, then X determines Y and Z separately.

$X \rightarrow YZ$
$X \rightarrow Y$
$X \rightarrow Z$

6) Pseudo Transitive Rule → If X determines Y and YZ determines W, then XZ determines W.

$X \rightarrow Y$ and
$YZ \rightarrow W$
$XZ \rightarrow W$

Q. Find the candidate key of relation R(A, B, C, D, E, F) with F.D.

$A \rightarrow C$
$C \rightarrow D$
$D \rightarrow B$
$E \rightarrow F$

Sol: $A \rightarrow C \rightarrow D \rightarrow B$ $\quad E \rightarrow F$

$\{AE\}^+ \rightarrow \{A, B, C, D, E, F\}$

b) R(A, B, C, D) $\quad A \rightarrow B \quad B \rightarrow C \quad C \rightarrow D$

candidate key = AD

Normalization :-

* It is the process of organizing the data in database.
* It is used to minimize the redundancy from a relation or a set of relation.
* It is used to eliminate undesirable characteristics like insertion, update, deletion anomalies.
* It divides larger table into smaller and links them using relationship.
* The normal form is used to reduce redundancy from the database table.

Types of Normal Form (Normalization) →

i) 1 NF → * A relation is in 1NF, if it contains an atomic value.
* It eliminate repeating groups.

ii) 2 NF → * A relation will be in 2NF, if it is in 1 NF and all non-key attributes are fully F.D. on the primary key.
* It eliminate partial F.D.

iii) 3NF → * A relation will be in 3NF, if it is in 2NF and no transition dependency exists.
* It eliminate transitive dependency.

iv) BCNF → * Boyce Codd's Normal Form.
* A Strong definition of 3NF is known as BCNF.
* It is also kndwon as 3.5NF.

v) 4NF → * A relation will be in 4NF, if it is in BCNF and has no multi-valued dependency.
* It eliminates multi-valued dependency.

vi) 5NF → * A relation is in 5NF, if it is in 4NF and does not contain any join dependency, joining should be lossless.
* It eliminates join dependency if any.

Examples:-

1NF

Student Table

S_ID	S_name	S_Phone no	S_Branch
11	Kamal	7002412314	CS
		9826245689	
12	Karan	8004829412	IT
		9685364491	
13	Ravi	6362581492	CS
		7004812591	
14	Ram	9823495681	IT
		7088828121	
15	Kamal	6824138412	CS
		9823768912	

Decomposition

11	S-name	12
11	S-phon	12
11	S-Phon	12
11	S-Branch	12

15
15
15
15

2NF

Order No.	Title	Qty	Unit Price
1	N/w	1	250
1	Java	1	275
2	DBMS	2	225
2	Multimedia	1	300
2	Data Str.	1	190
3	Multimedia	1	295
3	DBMS	2	300
3	N/w	5	250

** Order No, title → Qty ∴ Fully F.D

Order no. ↛ Qty
title ↛ Qty

Order Master Table

Order	title	Qty

Price Master Table

Title	Price

3NF

Emp_no	Emp_name	Salary	Dept_name	Dept_location
1001	Vishal	80,000	Account	102
1002	Amit	90,000	Sales	104
1003	Anuj	75,000	Accounts	102
1004	Vikas	85,000	Sales	104
1005	Sumit	72,000	Stores	106

✗ Emp_no → Dept_name → Dept_Location
✗ Emp_no → Dept_location

Decomposition

Emp_no.	Emp_name	Salary	Dept_name

Dept_name	Dept_location
Accounts	102
Sales	104
Store	106.

BCNF

Emp_ID	Emp_country	Emp_Dept	Dept_type	Emp_dept_No
265	India	Designing	D 304	283
264	India	Testing	D 394	300
364	UK	Store	D 283	232
364	UK	Developing	D 293	549

Candidate key

Emp_ID → Emp_country
Emp_Dept → {Dept_type, Emp_dept_no}

country table

Emp_ID	Emp_country
265	India
264	"
364	UK.

dept. table

Emp_dept	Dept_type	Emp_dept_No
Design.	D 304	283
Test	D 394	300
Store	D 283	232
Develp.	D 293	549

After Mapping.

Emp_ID	Emp_dept
265	Designing.
264	Testing
364	Store
364	Developing

4NF

faculty	Subject	committee
Vikram	DBMS	Placement
"	Java	"
"	C	"
"	DBMS	Scholarship
"	Java	"
"	C	"

Decomposition

Subject Table

faculty	Subject
Vikram	DBMS
"	Java
"	C

Committee Table

faculty	Committee
Vikram	Placement
"	Scholarship

5NF

- Also known as PJNF
  - Project → join
- Decompose smaller table.
- All tables broken in many table as possible in order.
- If it is equal to original table then 5NF.
  table 1 + table 2

faculty	Subject	committee
Vikram	DBMS	Placement
"	Java	"
"	C	"
"	DBMS	Scholarship
"	Java	"
"	C	"

Date 18th April

Commands :-

* system cls → To clear the screen
* Create database my_database;
* create table emp_cdac;
* insert into emp_cdac (emp_id, emp_name, emp_sal) values (101, "Mitali", 90000), (102, "Saurabh", 95000);
* alter table emp_cdac ADD COLUMN emp_dept VARCHAR (50);
* use database;
* create table emp_cdac (emp_id INT Primary key, emp_name VARCHAR (50), emp_salary DECIMAL(10,2));
* update emp_cdac SET emp_dept = "Admin" where emp_id = 101;
* truncate table emp_cdac; → delete data from table.
* drop table emp_cdac; → delete table completely

To change data type

* alter table table_name modify column column_name datatype ( );
* delete from table_name where col_name = value;
  " " grades where st_id = 1;

Transaction Properties :-

* Collection of operations that a form of single logic unit of work is called transaction.
* It is the unit of program execution that accesses and possibly updates various data items.
* It is defined as a logical unit of database processing that includes one or more database accesses operation.

There are two types of operations :-

i) Read (X) → It is used to read the values of account 'X' from the database and store it in a buffer in main memory.

ii) Write (X) → It is used to write the values back to the database from the buffer.

Ex (1) To debit transaction from an account which consist of following operation.
X → 1000, Y = 1000

1) R(X)
2) X = X - 500
3) W(X)

T_1	T_2
read(x) - 1000	read(Y)
X = X - 500	Y = Y + 500
write(X)	write(Y)
X = 500	Y = 1500

(2) If given transaction, the debit fails after execution operation 1 then X values will remain 400 in the database which is not expe. accepted by bank. How to solve this query?

Ans To solve this problem, use commit and roll back.
commit - to save the work permanently.
roll back - ~~start~~ from where transaction fails.

ACID Properties :-

Atomicity → Abort, Commit
Consistency → Before, After
Isolation → Single Operation runs at a time
Durability

Atomicity → It states that all operation of the transaction takes place at once.
* If not, the transaction is aborted.
* If the transaction is abort then all the changes may not possible (visible).
* If a transaction commits, then all the changes are possible (visible).

Consistency → The integrity constants are maintained so that the database is consistant before and after the transaction.
* The execution of a transaction will leave a database in either its prior stable state or a new stable state.

* It's property of database states that every transaction sees a consistant database instance.
* It is also used to transform the database from one consistant state to another consistant state.

Eg: Let us assume that following transaction T consisting of T_1 and T_2. A consist of 'A' consists Rs 600 and 'B' consists Rs 300. Transfer Rs. 100 from A/c A to B.

T_1	T_2
Read (A)	Read (B)
A = A − 100	B = B + 100
write (A)	write (B)
A = 500	B = 400

Isolation → It shows that the data which is used at the time of execution of a transaction cannot be used by the second transaction until the first one is completed.
* Concurrency control sub-system of DBMS enforced the isolation property.
* ~~This property is used~~

Durability → It indicates the performance of the database in consistant state.

* It states that the transaction made the permanent changes.
* They cannot be lost by the erroneous operation of the faulty transaction or by the system failure. When a transaction is completed then the database reaches a state known as consistant state. That consistant state cannot be lost even in the event of system failure.
* The recovery subsystem of a DBMS has responsibility of durability property.

19th April
Date Page

SQL Data Types

- Numerical DT
  - INT or INTEGER
    - range { -2147483648 to 2147483647 }
  - BIGINT
    - range { -9223372036854775808 to 9223372036854775807 }
  - Float
  - Double or Real
- String DT
  - CHAR
  - VARCHAR {255}
  - TEXT
  - TINYTEXT
  - MEDIUMTEXT
  - LONGTEXT
- Date & Time DT
  - Date { YYYY-MM-DD }
  - time { HH:MM:SS }
  - Datetime { YYYY-MM-DD HH:MM:SS }
  - TIMESTAMP

Other Data Types

- Boolean or Bool
  - TINYINT(1) { 0 - false, 1 - true }
- ENUM : (storing predefined list of values)
- SET (Similar to ENUM)
- BLOB (Storing binary large object like images, files)

Date Page

INT → Storing integer values
BIGINT → Storing large integer values
Float → Storing approximate numeric value with decimal point
CHAR → Storing fixed length string
VARCHAR → Storing variable length string
TEXT → Store large amount of text.

SQL Constraints

Primary key → Unique identification of records.

foreign key → Establish relations b/w Tables

Unique → It enforce uniqueness on column values.

check → To enforce specific condition on column values.

NOT Null → To ensure that columns cannot be null values.

Default → To specify default values of columns.

Index → To create indexes for faster data retrieval.

Avg() → compute the avg value in a col^n.
select AVG(age) from table_name;
col^n name

30 April

Functions

Aggregated function :-

Max() → Retrieve the max value in a col^n
select Max(col_name) from table_name

Min() → Retrieve a min value in a col^n
select min(col_name) from table_name

Count() → The no. of rows or null values in a column.
select count(*) from table_name;

SUM() → Calculate the sum of values in a column.
select SUM(col^n_name) from table_name;

String function :-

Concat() → concatinates two or more strings.
select CONCAT (col_1, col_2) AS new_col from table_name

Upper() / Lower() → convert a string into UPPERCASE/LOWERCASE.
select UPPER/LOWER (col^n_name) from table_name;

Mathematic function (Basic Arithmatic) :-

Addition	+	select 2+7 AS SUM;
Subtraction	-	select 2-7 AS Diff;
Division	/	select 2/7 As div;
Modulus	%	select 2%7 As mod(remainder);
Multiplication	*	select 2*7 As product;

Comparison function :-

Equal = → It check the value of operands are equal. It will returns true roles.
Not equal <> or != → select col_1, col_2 from table_name where col_1 != values;
Greater than >
Less than <
Not greater !>
NOT Less !<
Greater and equal >=
Less and equal <=

select st_name, st_age from student where st_age >= 18;

Eg: Calculate total price including tax.

Query:- select order_id, total_amount * 1.15 AS Total_Price_with_tax from table_name ;

Operators

Logical Operator :-

AND → It is used to combine multiple operations (conditions) and it returns true only if all the conditions are true.

Select col^n_1, col^n_2 from Table_name where condition_1 AND condition_2 ;

Eg: Select emp_name, dept, salary from Employee where dept = 'CS' AND salary > 50000 ;

OR → It is used to combine multiple condition and it returns true if atleast one of condition is true.

select col^n_1, col^n_2 from table_name where condition_1 OR condition_2 ;

Eg: Select emp_name, dept, salary from Employee where dept = 'CS' OR salary > 50000;

NOT → It is used to negate a condition returning true if condition is false and false if condition is true.

Select col_1, col_2 from table_name where NOT condition;

Ex Select emp_name, dept from Employees where NOT dept = 'Admin';

Combining Logical operator

Select emp_name, dept, salary from Employee where (dept = 'CS' AND Salary > 50000) OR dept = 'Admin';

LIKE → It is used to match patterns in string column which used wild card characters like (%) for 0 or more characters, _ (underscore) for a single character.

Select col_1, col_2 from table_name where col_1 LIKE "pattern";

Ex Select cust name from customer wher cust_name LIKE "VIBHANSHU%";

ALL, SOME (ANY) → It is used with sub-queries to compare a value to all or some value returns by the sub-queries.

Select col_1, col_2 from table_name where col_1 OPERATOR ALL (sub-query);

Ex Select prod_name from Products where prod_price > ALL (select prod_price from Products where prod_category = "Electronics");

EXISTS → It is used to test for the existance of rows return by a sub-query.

Select col_1, col_2 from table_name where EXISTS (sub-query);

Ex Retrieve orders that have at least one associated product.

Query Select order_id from orders where EXISTS (select 1 from order_items where order_items.order_id = order.order_id);

IN → It is used to check if a value matches any value in a list or sub-query result.

Select col_1, col_2 from table_name where col_1 IN (value 1, value 2, value 3);

Eg. Select cust_name from customer where city IN ('Gwalior', 'Bhopal', 'Delhi');

BETWEEN → It is used to specify the range of values.

Select col_1, col_2 from table_name where col_1 BETWEEN value_1 and value_2;

Eg. Select prod_name from Products where prod_price BETWEEN 50000 and 100000;

Compound Operators:—

+= (Addition Assignment) → It is used to add a value to an existing value and assign the result back to the column.

update table_name SET col_1 += value where condition;

Eg. Increase 5000 in Sales for everyone.

UPDATE employees SET salary += 5000 where department = 'Sales';

-= (Subtraction Assignment) → It is used to subtract a value from an existing value and assign the result back to the column.

UPDATE table_name SET col_1 -= value where condition;

Eg. Decrease the quantity of product by 1 for a specific product id

UPDATE products SET quantity -= 1 where product_id = 527;

*= (Multiplication Assignment) → It is used to multiply a value in an existing value and the result back to the column.

UPDATE table_name SET col_1 *= value where condition;

Eg. Increase the sale amount by apply the 10% commission rate.

UPDATE sales SET total_amount *= 1.10 where commission_rate = 0.1;

/= (Division Assignment) → It is used to divide a value in an existing value and result back to the column

UPDATE table_name SET col^1 /= value where condition;

Ex Adjust the avg. score of students by dividing it by 2 to scale it down.

UPDATE Student_score SET avg_score /=2 where grade level = 12th;

%= (Modulus Assignment) → It performs modulus operation on an existing column value with a specified value and result back to the column.

UPDATE table_name SET col^1 %= value where condition;

Ex Update the discount% for products to a new value based on modulus operation.

UPDATE product SET discount_percent %= 20% where category = 'Electronics';

&=, ^=, |=

Bitwise Operator :-

& (AND) → Value 1 & Value 2

It performs bitwise AND operation b/w corresponding bits of two integers.

Value 1 & value 2
Select 5 & 3;

| (OR) → It performs bitwise OR operation b/w corresponding bits of two integers.

Value 1 | value 2
Select 5 | 3;

^ (Ex-OR) → It performs bitwise EX-OR operation b/w corresponding bits of two integers.

Value 1 | Value 2
Select 5^3;

&= (Bitwise AND Assignment) → It performs a bitwise AND operation b/w the left hand operand and the right hand operand and then assign the result back to the left hand operand.

variable &= value;
col_name &= value;

~~Set @x=5~~ set @x = 5;
~~set @x=3~~ set @x &=3;

To show:-
Select @x; //Result is 1.

^= (EX-OR Bitwise Assignment) →

variable ^= value;
col_name ^= value;

set @ x = 5;
set @ x ^= 3;

Select @x;

|= (OR Bitwise Assignment) →

variable |= value;
col_name |= value;

set @x = 5;
set @x |=3;

Select @x;

Set Operation :- These below operators are used for combining and comparing the result of sets of two or more select queries.

Union, UnionAll, Intersect, Minus.

Union Operator → combine the result set of two or more select statements, removing duplicates from the combined result set.

Select col_1, col_2 from table_name1 UNION
Select col_1, col_2 from table_name 2;

Eg. Select emp_id, emp_name from Employees UNION
Select cont_id, cont_name from contractors;

UnionAll Operator → It is similar to union but it retains all rows from the result sets of the component select queries including duplicates.

Select col_1, col_2 from table1 UNIONALL
Select col_1, col_2 from table2;

Eg. Select emp_id, emp_name from Employees UNIONALL
Select cont_id, cont_name from contractor;

Intersect and Minus does not use in Mysql. We use Innerjoin and ~~exists~~ left join instead of them resp.

JOINS

Inner Join → It returns rows from both tables (like table1 and table2) that have matching values in the specified column.

Practice

Join two tables through student id in grades or id in students

Students table

id	name	age	gender
1	Alice	28	f
2	Bob	22	m
3	Charlie	21	m
4	Diana	23	f

Grades table

id	st_id	course_id	grade
1	1	101	A
2	1	102	B
3	2	103	A-
4	3	101	B+

```
select Students.name, Grades.grade from Students
INNER JOIN Grades ON Students.id=Grades.st_id;
```

~~INNER JOIN~~

This query will fetch the names of students, grades for courses and courses in which they enrolled.

* INNER Joins only includes rows that have matching values in both tables.

Left Join → It returns all rows from the left table (Students) and matching rows from the right table (grades). If there is no match, NULL values will be returned from right table.

```
Select Students.name, Grades.grade from Students
LEFT JOIN Grades ON Students.id = Grades.st_id;
```

Students Table

id	name
1	Alice
2	Bob
3	Charlie
4	Diana

Grades Table

id	st_id	grade
3	2	A-
4	3	B+
5	4	A
6		

O/P

name	course	grade
Alice	NULL	NULL
Bob		
Charlie		
Diana		

Right Join → It returns all rows from the right table (grades) and matching rows from the left table (students). If there is no match, NULL values will be return from for left table.

```
Select Students.name, Courses.course_name,
Grades.grade from Students RIGHT JOIN
Grades ON Students.id = Grades.st_id
RIGHT JOIN Courses ON Grades.course_id =
course_id;
```

Cross Join → It returns cartesian products of two tables resulting in all possible combination of the row from both tables.

```
Select Students.name, Courses.course_name
from Students CROSS JOIN Courses;
```

Self Join → It is used to join a table with itself, typically to find related record within the same table.

Add some data in studat table then,

```
select S1.name AS Student1, S2.name AS Student2
from Students S1 JOINS Students S2 ON
S1.age = S2.age AND S1.id <> S2.id;
```

22nd April

Create a user-defined function named CalculateDiscount that accepts the price of a product as input and returns the discounted price after applying a 10% discount.

```
Delimiter //                    Set the delimeter '//' for temperory
                                to allow definition multi-line
                                fun bodies.

Create function CalculateDiscount(price) Decimal(10,2))
Return Decimal(10,2);
                                Deterministics
                                (return same value (I/p))
                                No SQL (same format execute)
Begin
    Declare discount_price Decimal(10,2);
    Set discount_price = price * 0.9;
    Return discounted_price;
End //
Delimeter;

Select product_name, price, CalculatedDiscount(price)
AS discounted_price from Products;

Select CalculateDiscount(100);
Select CalculateDiscount(products.price) from Products;
Select Products.*, CalculateDiscount(products.prod_unit_price)
As prod_finalPrice from products;
```

View and Indexing

View :- It is a virtual table that is dynamically generated from the result of a select query.

* They do not perform insert, update, delete
* It is used for generating reports, joining queries

Syntax:- create view view_name AS select col1, col2 from table_name where condition;

Ex. Create view Employee_view AS select emp_id, emp_name, dept_id from Employees where dept_id = 1;

Select * from Employee_view;

Advantages →

i) Simplify queries
ii) Data abstraction
iii) Security control

Uses of View :-

i) Performance
ii) Updatability → read only
iii) Maintainance

Index :- It is a data structure associated with a table that allow for efficient retrieval of rows based on the values of certain columns.

* It performs select, update, delete and join operations.

Types of Index →

i) Primary key Index → Uniquely identifies each row in the data table.

for ex:- create table Users (user_id INT Primary key, user_name VARCHAR(50), email VARCHAR(50));

ii) Unique Index → Unlike a primary key a unique index contains null values. (Multiple rows can have null in a unique column).

* It ensures that the values in the index column are unique across the table.

for ex:- create table Products (id INT Primary key, name VARCHAR(50) Unique, price DECIMAL(10,2));

iii) Index → It can be created one or more columns to improve query performance.
It performs insert, update, delete operations.

for ex:- create table Orders (id INT Primarykey, cust_id INT, date DATE, total_amount DECIMAL(10,2), INDEX idx(cust_id));

iv) Composite index → It consists of multiple columns. It is useful for queries that filter or sort by multiple columns simultaneously.

Advantages →

i) Improve query performance
ii) Faster sorting and joining operations.
iii) Constraints enforcements

Uses of Index →

i) Index maintainance → insert, update, delete operation performed

ii) Index selection →

iii) Index cardinality → unique values

23rd April

Clustering Index :- → In database, recorders the way records in the tables are physically stored.
→ In each table have one index of the rows. Because the physical ~~record~~ order of the rows is known by the cluster.

key Points:-
★ Leaf Node → Actual table data page
★ Primary key in the SQL server, automatically create a cluster index unless specified.

for ex:- create a cluster index on the emp_id column of a table will physically reorder the rows of the tables on the emp_id values

Syntax:- create Clustered Index idx_emp_id ON employes(emp_id);

Non-Clustering Index :-

★ It is a separate structure from the actual table data that stores a ~~it~~ sorted list of reference to the table's rows.
★ It does not effect the physical order of the rows in the table.

key Points:
★ Multiple non-clustered index can be created on the table.

for ex:- Create a non-cluster index on the last_name col^ of the table will create a separate data structure that holds reference to rows based on the last_name.

Syntax:- create index idx_name on Users last_name ;
(idx_name, col'name, table name)

Covering Index :-

* In this type of index that includes in all columns needed to satisfy a query thereby allowing the query to be executed using only the index without accessing the table.

key Points:-

* It reduce i/p, o/p operation because the necessary data is retrieved directly from the index.
* It is useful for queries that involves multiple columns or required extensive data retrieval.

Syntax:- create index idx name on table_name (col'name) includes (first_name, last_name) ;

Stored Procedure :-

* It is a named set of SQL statement that can be executed repeatedly.

Syntax:-
```
Delimiter //
Create Procedure procedure_name (table_name, col'name)
Begin
    .... SQL statement
End //
Delimiter ;
```

* It uses because of complex logic
* reducing redundancy
* Improving performance by executed
* Pre-defined SQL operation.

for ex:-
```
Delimiter //
Create procedure getCustomer ( IN cust_id, id INT)
Begin
select * from Customer where Id = cust_id ;
End //
Delimiter ;
```
(IN operator (to store data in table))

Execution → call getCustomer (1); (1 ← cust_id.)

* Insert, update, delete operations used in it.
* A collection of pre-compiled SQL statements stored inside the database.
* It is created to perform one or more DML operations on database

Table (1) Employees

Emp_id	name

Table (2) Departments

dept_id	name

```
Delimiter //
Create Procedure getEmployeeByDepartment (IN dept_id INT)
BEGIN
    Select e.* , d.dept_id
    from Employees e
    JOIN Departments d ON e.emp_id = d.dept_id
    Where d.dept_id = dept_id;
END //
Delimiter ;
```

24th April

Trigger :-

* It is a special type of stored procedure that is invoke automatically in response to an event.
* Each trigger associated with a table which is activated on any DML statement such as insert, update, delete.

Syntax :-
```
Create trigger trigger_name Before/After insert/update/delete on table
for each row
Begin
    ... SQL statement
End;
```

for eg :-
```
Delimiter //
Create trigger update_stock
After insert on Orders
for each row
Begin update Products
    set stock = stock - new.quantity where id = new.prod_id;
End; Delimiter;
```

Cursor :-

* It holds multiple rows received by a SQL statement.

Syntax :-
```
Delimiter //
Declare cursor_name Cursor for select_statement;
Open cursor_name;
fetch cursor_name into variable;
    ... process row here
close cursor_name;
Delimiter;
```

```
DELIMITER //
CREATE procedure display_customer_name()
BEGIN
    DECLARE done INT Default false;
    DECLARE customer_name VARCHAR(50);
    DECLARE cur cursor for
        select name from customers;
    DECLARE continue handler for not found
        set done = true;
    OPEN cur;
    read_loop: loop
        fetch cur into customer_name;
        if done then
            leave read_loop;
        End if;
        select customer_name;
    End loop;
    CLOSE cur;
END //
DELIMITER;
```

(variable)

it can be skipped → ×print it (process of each customer name)

Mongo DB

It is document design for easy development and scaling.

Mongo → It is a command line shell that connect to a specific instance of MongoD.

MongoD → It is basically the host process for the database.
(Daemon)

```
{ fields          values
  name: " ",
  age:
  status: "A",
  group: ["Acts", "Project Engineer"]
}
```

SQL	MongoDB
Database	Database
Tables	Collection
rows	Document (BSON)
columns	field

BSON → Binary Java Script Object Notation

Show databases
use db_name ← this is new database
-- switch to db_name

```
{
  name: "___",          // To create collection
  age: ___,
  group: "___"
}
```

// after this db_name database will be created.
// After that this new database will be shown in existed databases with fields

Syntax :-

db.dropDatabase()

→ Collection :-

Show collection
db.create collection ("Comment")
db. Comments. drop()

→ Row command :-

db . comments. find ( )
db. comments. find ( ). pretty( )
↳ predefined

→ Update :-

db. comment. update ({ name: "Ant" },
{ name: "ans",
lang: "Javascript",
member: 51 },
{ upsert: true })

→ Increment Operator :-

db. comments. update ({ name: "Vikram" },
{ $inc: { member: 2 } })

→ Rename Operator :-

db. comments. update ({ name: "Vikram" },
{ $rename: { member: "members" } })

→ Delete Row :-

db. comments. remove ({ name: "Vikram" })

→ Less than / Greater than / Less than or Equal / Greater than or Equal
(lt) (gt) (lte) (gte)

db. comments. find ({ member : { $ lt : 90 } })
db. comments. find ({ member : { $ lte : 90 } })
db. comments. find ({ member : { $ gt : 90 } })
db. comments. find ({ member : { $ gte : 90 } })