12/12/19

Introduction to DBMS:

A Database is a collection of related data which represents some aspects of real world.

→ A database system is designed to be built and populated with data for certain task.

→ DBMS basically consist of set of inter related data and set of programs for accessing those data.

→ The primary objective of DBMS is to provide an environment i.e not only convenient but also efficient to use while retrieving and storing database information.

**Database system Applications:-

→ DBMS are used either directly or indirectly by many applications

→ Applications of DBMS are:

i) Bank transactions
ii) Ticket Reservation
iii) students at universities
iv) Database in a library
v) Shopping at Super Market

vi) Telecommunications
vii) Internet interactions (online shopping)
viii) Airlines (reservations, schedules)
ix) Sales (customers, products, purchase)

NOTE:- DBMS is a software that perform a following functions

i) defining a database
ii) Solving the data
iii) Supporting query Language
iv) producing reports
v) creating data entry screens

13/12/19

File Management System:-

One way to keep the information on a computer is to store it in permanent system i.e Files.

→ It is used to store logically related records permanently.

→ In this system each record has a separate file

Drawbacks of file Management System.

→ Data Redundancy:- as various copies of same data resides indicating the redundancy of data which leads to higher storage and access cost.

→ data inconsistency:- As a same data resides in many different files so the data inconsistency increased.

→ difficulty in accessing data :- need to write a new program to carry out each new task.

→ data isolation:- Multiple files and formats, so it is difficult to write new programs to retrieve the appropriate data.

→ Integrity problem:- Integrity ~~constants~~ constraints become part of program code. hard to add new constraints or change existing once

→ Atomicity problem:- failures may leave data base in an inconsistence state with partial updates carried out.

→ concurrent access by multiple users:- concurrent access needed for performance where uncontolled concurrent access can lead to inconsistancy.

+ Security problem:- Every user is able to access all the data from everywhere as various copies of same data exists

Purpose of database system is solution for all above problem.

View of data:-

A database system is to provide users with an abstract view of data.

Data Abstraction:

i) physical level :- The lowest level of abstraction describes how the data are actually stored.

ii) logical level:- The next higher level of abstraction describes what data are stored in database and what relationship exits between those data.

→ database administrator, who must decide what information to be keep in database

iii) view level:- The highest level of abstraction describes only part of entire database.

→ Many users do not need all these information they only need to access

a part of the database.

→ The view level of abstraction exists to simplify their interaction with the system.

→ The system may provide many views for the same database

Instances & Schemas:-

→ The collection of information stored in the database at a particular movement is called instance of database

→ The overall design of database is called the database schema.

→ database design have different schemas

i) physical schema

ii) logical schema

iii) external schema (sub schema)

The ability to modify a schema definition in one level without effecting a schema definition in the next higher level is called data Indepence.

Two levels of data independence

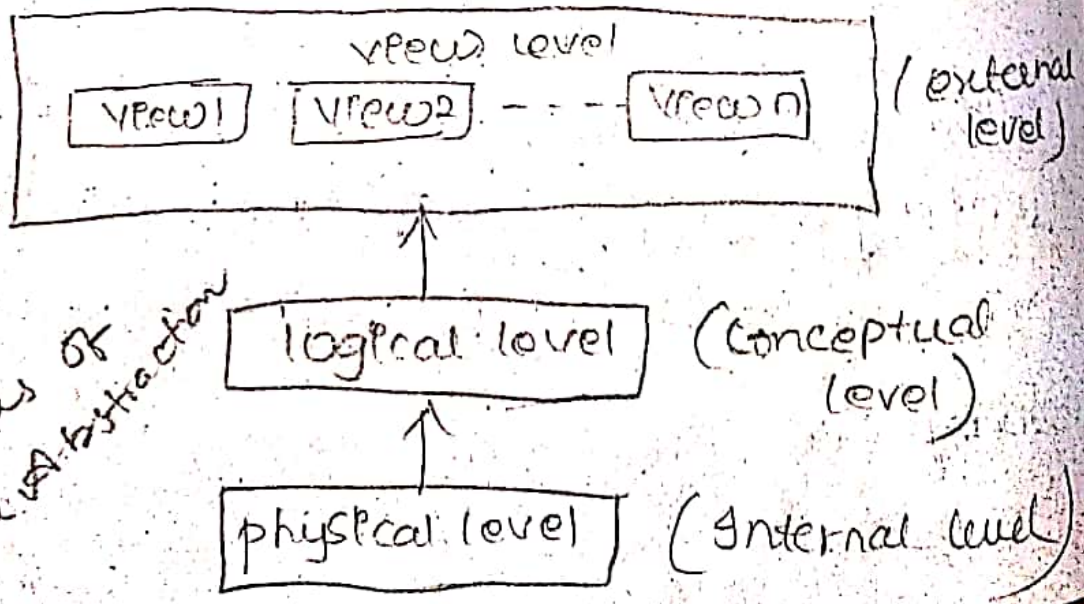
i) physical data Independence :- It is the ability to change the physical schema without the applications programs to be rewritten.

→ changes made to external schema without affecting the conceptual or logical schema

→ modifications are done on storage structure such as storage devices, Indexing, Switching etc.

ii) logical data independence : It is the ability to modify logical schema which decides what information is to be kept in data base without affecting next level schema

→ modifications are done on data structures such as entities, attributes and relationships



Database Languages:

→ A database system provides appropriate languages & interfaces for each category of users to express database queries and updates.

→ Database Languages are used to create & maintain database on computer.

→ A Database system must provide a data definition language to specify the database schema and data manipulation language to express database queries & updates.

→ There are large no.s of database languages like Oracle, MySQL, MS Access, dBase, foxpro.

→ SQL statements commonly used in Oracle and MSAccess can be categorized as DDL, DCL, DML (Definition, Control, Manipulation

(i) Data Definition Language (DDL):

DDL is used to specify the database schema for database system

It is a language that allows the users to define data & their relationship to other types of data

It is mainly used to create files, database, data dictionary and tables within database.

It is also used to specify the structure of each table, set of associated values with each attribute, integrity constraint, security & authorization.

* domain Constraints (column condition):
It is the most elemental form of integrity constant.
Ex: Integer type, Character type, date(or) time types.
They are tested easily by the system whenever a new data item is entered into the database.
A domain of possible values must be associated with every attribute.

* Reffernecial Integrity: To ensure that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relationship (foreign key).

* Assertions: Domain constraints and reffernecial Integrity constraints are special forms of assertion. It is any condition that data base always satisfy.

* Authorization: Authorization to read data it means the system giving the access to read but not to do any modifications, deletions, or insertions.

* Output of DDL is placed in data dictionary which contains metadata (data about data)

The following table gives an overview of DDL statements in SQL

i) CREATE: It is used to create the database or its objects. like (table, index, function, views store procedure and triggers)

ii) ALTER: It is used to alter the structure of the database. (add, delete, modify - columns)

iii) DROP: It is used to delete the objects from the database.

iv) RENAME: It is used to rename an object existing in the database.

v) DESCRIBE (or) DESC: It is used to describe the structure of database.

14.12.19

(ii) Data Manipulation Language (DML):

It is a language that provides a set of operation to support the basic data manipulation operations on the data held in the database

-> It allows user to insert, update, delete, and retrieve data from the database.

-> A Query is a statement requesting the retrieval of information. The part of DML that involves data retrieval is called a Query Language.

-> Select command is a Data Query Language (DQL)

Two types of DMLs

i) procedural DMLs: Require a user to specify what data are needed and how to get those data. It is Relational Algebra.

ii) Non-procedural DMLs: It is also known as declarative DMLs. It require a user to specify what data are needed without specifying how to get it.

It is more easier to learn than procedural DML.

→ It is nothing but our SQL.

The following table gives an overview of DML statements in SQL

i) INSERT: Add new rows of data into table, or view

ii) UPDATE: Change column values in existing rows of a table or view. i.e, update existing data within a table.

iii) DELETE: Remove records from tables or views

iv) SELECT: Retrieve data from one (or) more tables.

(iii) Data Control Language (DCL):

This statements control access to data of database using statements such as GRANT and REVOKE

The following table gives an overview of DCL statements in SQL are:

i) GRANT: Gives user's, access privileges to database.

2) REVOKE: We draw access privileges given with the ~~grant~~ GRANT command.

(IV) Transaction Control Language (TCL):

TCL statements are used to manage the changes made by DML statements. It allow statements to be grouped together into logical transactions.

The following table gives an overview of TCL stmts. in SQL are:

1) COMMIT: Save work done.
2) ROLLBACK: Restore database to original since the last commit. (Rollback's a transaction incase of any error occurs).
3) SAVEPOINT: Identify a point in a transaction to which you can later rollback.

* Relational Databases: A Relational database is based on the relational model & uses a collection of tables to represent both data & relationships among those data. It also includes a DML and DDL statements.

i) TABLES: Each Table has multiple columns, and each column has a unique name. No duplicate names aren't allowed.

The Relation model is an example of a record based model.

ii) DML: The Query Language of SQL is non procedural. It takes as input several tables & always returns a single table.
ex: 'select' command.

iii) DDL: Execution of the DDL stmt creates the table. it updates the data dictionary which contains meta data. The schema of a table is an ex. of meta data.

iv) Database Access from App. programs:

* Application programs are the software codes that enable a user to interact with the database

→ The Application programs access the database in two ways
  i) By using Application programming Interface (API)
  ii) By extending the host language syntax.

i) Application Programming Interface:
It is a set of procedures that sends the DML and DDL statements to the data base and retrieve the result after processing from the database. The commonly used API include ODBC (Open database connectivity) that provides an interface for app.n programs written in C. and JDBC (Java

database connectivity) that provides an interface for application programs written in Java.

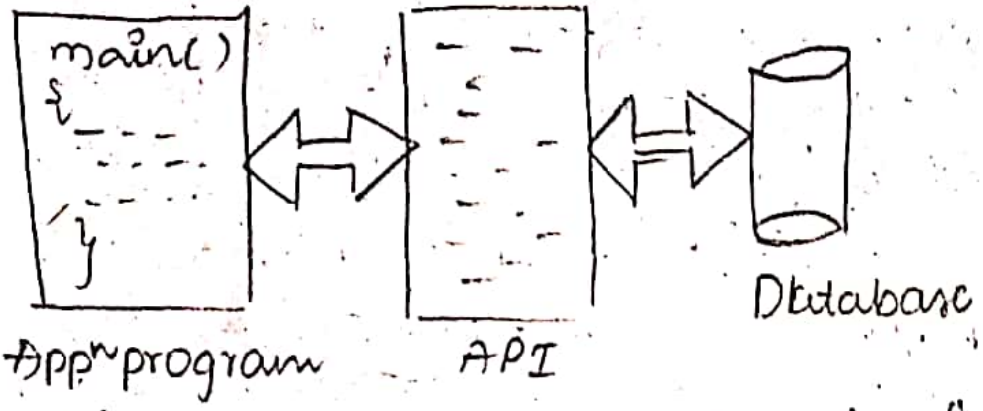


fig: Appⁿ program Accessing the database through API

ii) Extending Host Language Syntax:

```
COBOL stat
- - - - - - - -
- - - - - - - -
EXEC SQL
Select *
from emp
where
Emp_name: "John"
END-EXEC
- - -
- - -
```

Database can also be accessed by extending host language syntax so as to embed DML statements within the host programming language.

SQL which is a database sub language is embedded in a programming language. DML pre compiler converts the DML stmts

to normal procedural calls in the host language. It is use to compile the source code. The execution of this SQL embedded statements is different from normal execution.

* Data storage and Querying Querying:

-> The functional components of database system can be broadly divided into the storage managers & the Query processor components.

-> The storage manager is important because database require a large amount of storage space. (It cannot be stored in mm, so the information is stored in disk.)

-> The Query processor is important because it helps the database system simplify and facilitate access to data.

1) Storage Manager: A Storage Manager is a program module that provides the interface between the low-level data stored in the database and the application programs & Queries submitted to the system.

-> It is responsible for the interaction with the file manager.

-> The storage manager translates the various DML statements into low-level file system commands for storing, retrieving & updating data in the database.

The storage manager components include:

i) Authorization & Integrity manager: which tests for the satisfication of integrity constraints & checks the authority of users to access data.

ii) Transaction Manager: which ensures that the database remains in a consistent state instead of system failures.

iii) File Manager: which manages the allocation of space on disk storage & the data structures used to represent information stored on disk.

iv) Buffer manager: which is responsible for getching data from disk storage into main memory & handles data sizes on database.

The storage manager implements several datastorage:

Data files, which stores the database
Data dictionary, which stores metadata about the structure of the database
Indices, which can provide fast access to data items.
(Eg: Index in TB).

2) The Query Processor:

The Query processor components include:

i) DDL interpreter: which interprets DDL stmts and records the definitions in the data dictionary.

ii) DML compiler: which translates DML statements in a query language into an evaluation plan consisting of low-level instruction (Query Evaluation Engines).

→ The DML compiler also performs query optimization which picks the lowest cost evaluation plan from among the alternatives

iii) Query Evaluation Engine: which executes low-level instructions generated by the DML compiler.

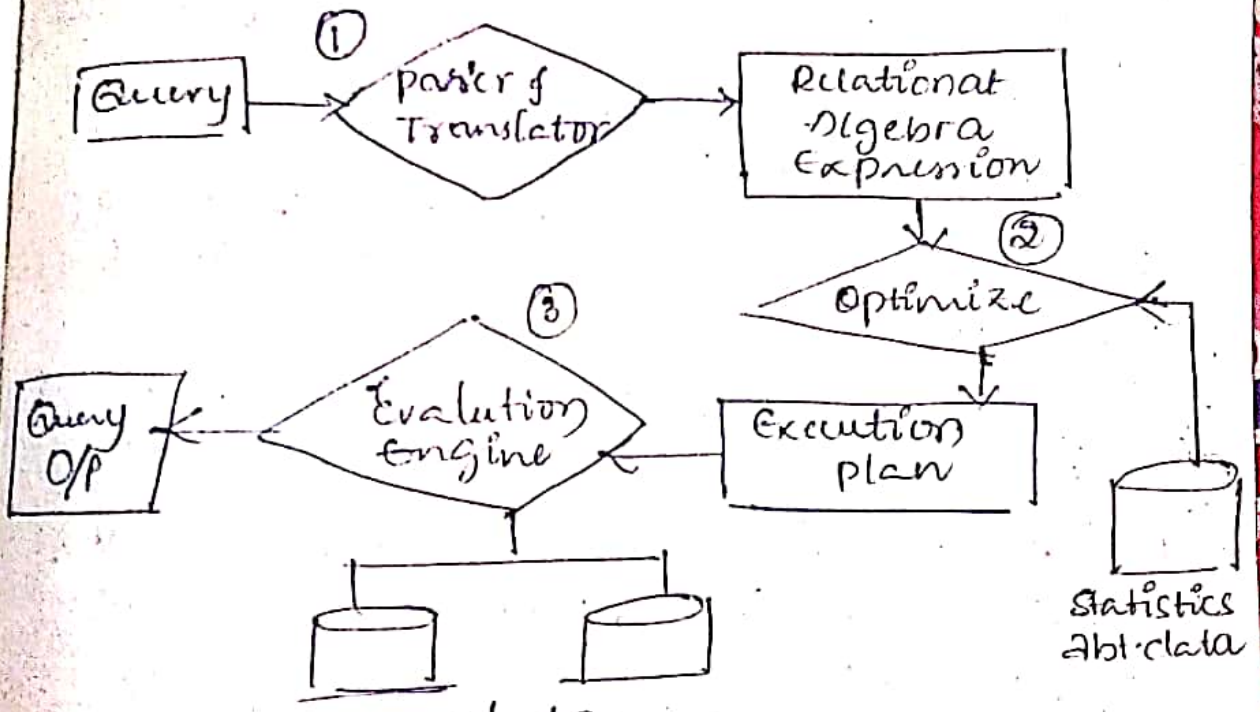


Fig: steps in Query processing.

21/12/19

Transaction Management:

A Transaction is a collection of operations that performs a single logical fn in a database application.

Each Transaction is a unit of both atomicity and consistency.

All (or) none requirement is called atomicity.

→ Durability: After a transaction completes successfully the changes it has made to the database persist even if there are system failures.

→ Isolation: The result of concurrent executions must be same as the result of the serial execution.

→ If the database was consistent, when a transaction started then the database has to be consistent when the transaction succesfully terminates

→ Transaction Management component ensures the atomicity and durability properties.

→ failure recovery: It detects system failures and restore the database to the state that existed previous

→ concurrency control Manager: Takes the responsibility to control the interraction among the concurrent transaction & ensure the consistency of database.

Database Administrators & Users:

* Database Administrator (DBA):- A person who works as central control over the system is called a DBA.

The functions of a DBA include

i) Schema definition.

ii) Storage Structure & Access method definition

iii) Schema and physical Organization modification.

iv) Granting of Authorization for data Access.

v) Routine Maintenance: As like backing of the database either on tapes or remote servers, monitoring jobs running on the database, ensuring the enough free disk space is available for normal Operation.

* Database Users: four different types of database system users

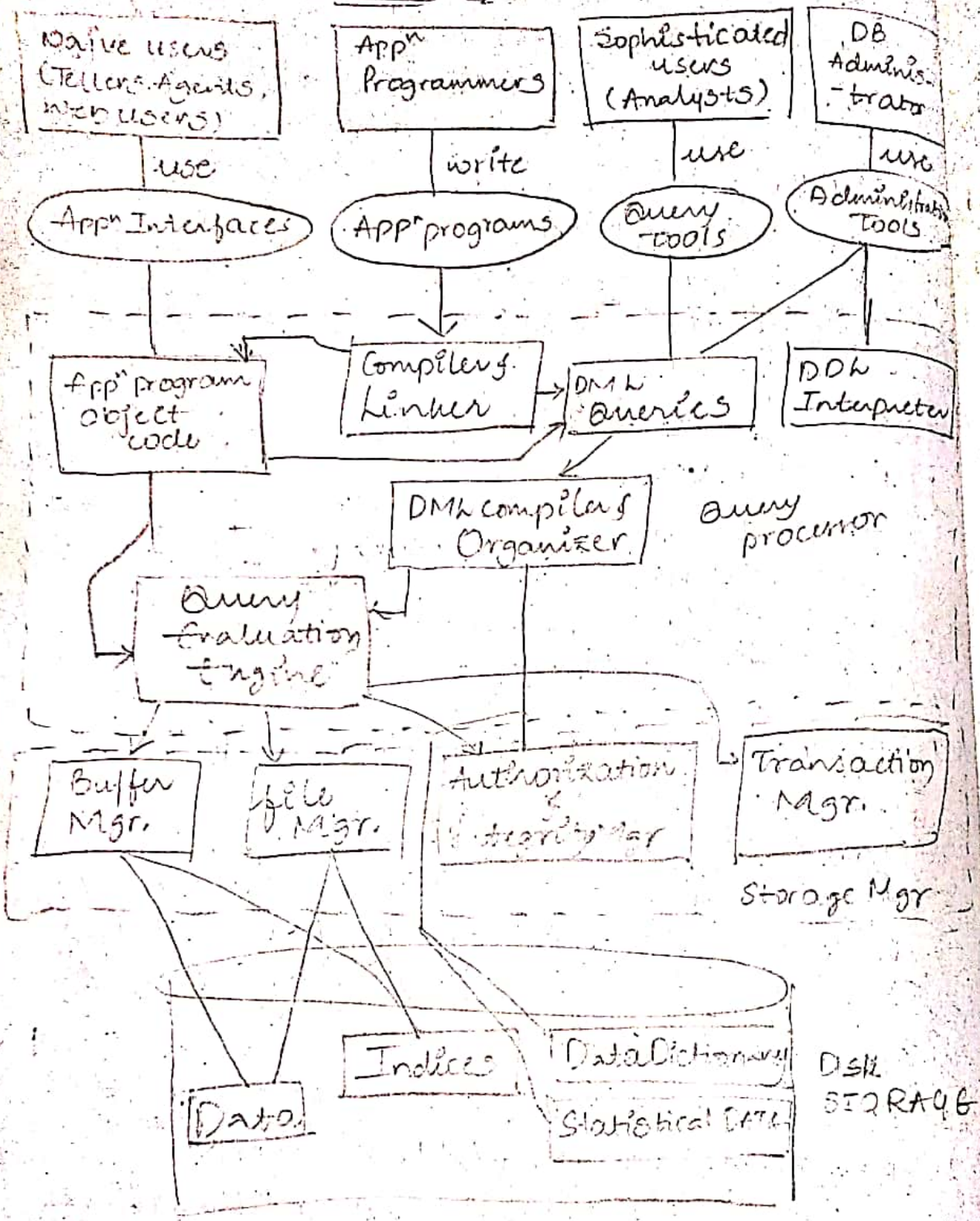
i) Naive Users:- Naive users are unsophisticate users who interact with the system by invoking one of the app^n programs that have been written previously.

ii) Application programers: They are computer professionals who write application program (for ex: RAD tools)

iii) Sophisticated Users: Interact with the system without writing programs (ex: Query Language)

iv) Specialized Users: They are sophisticated users who write specialized database app^{n}s that do not fit into the traditional data processing framework (EX: CAD Computer Aided Design)

v) Database Architecture:



Data base systems can be centralised or client server.

In a two tier architecture the app^n is partitioned into a component that resides

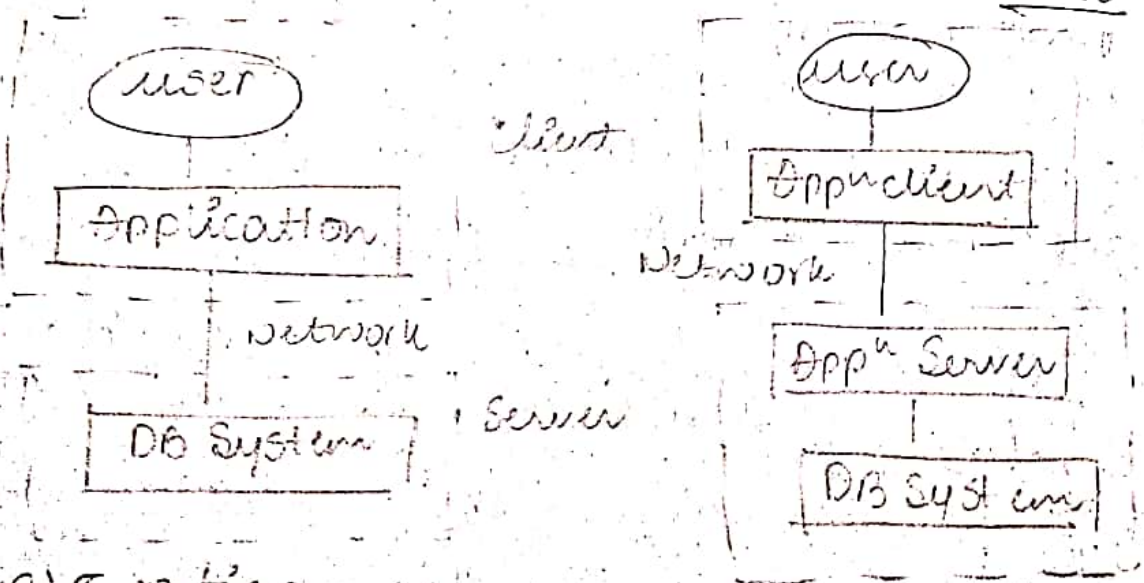
at the client machine. In this the front end directly communicates with a database running at backend.

→ API standards like ODBC and JDBC are used for interaction b/w the client and the server.

→ In a three tier architecture the client machine act as front end & doesn't contain any direct database calls. In this the backend divides into an app^n server and database server.

The client end communication with an app^n server through a server. The app^n server communicates with a database system to access data.

⇒ Three tier app^n are more appropiate for large app^n.s and for that app^n.s that run on www.



a) Two-tier Architecture

b) Three-tier Architecture

→ Database Design:

1) Design Process: The database designer needs to interact extensively with domain experts & users to carry out this task.

The designer chooses a datamodel, and by applying the concepts of the chosen datamodel translates these requirements into a conceptual schema of the data.

provide the detailed Overview of Enterprise.

What attributes & how ~~that~~ to group these attributes.

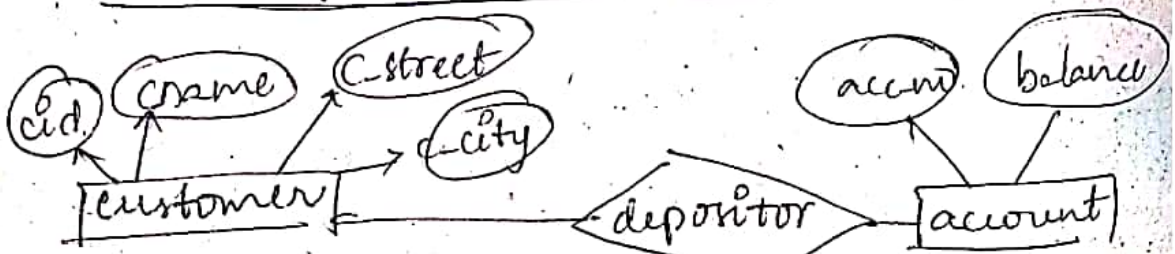
↳ Sol^{n} are 1) E-R model
2) Normalization.

→ In the logical-design phase, the designers maps the high-level conceptual schema onto the implementation data mode of the db system.

→ The designer uses the resulting system specific db schema in the subsequent physical-design phase, in which the physical features of the db are specified.

2) Db Design for Banking Enterprise

3) The Entity Relationship Model



E-R diagram

Rectangles
Ellipses
Diamonds
lines

4) Normalization

Datamining & Information Retrieval:

The term datamining refers loosely to the process of semiautomatically analysing large dbs to find useful patterns.

large companies have diverse source of data that they need to use for making business decisions. To execute queries efficiently on such diverse data, companies have built data warehouses. Data warehouses gather data from multiple sources under a unified schema, at a single site.

→ Querying of unstructured textual data is referred to as Info. Retrieval (based on keywords)

Specialty Databases:

1) Object-based data models: oop has become the dominant s/w development
   E-R model

2) Semi-Structured Data Models: It permit the specification of data where individual items of the same type may have diff. sets of attributes. Must have the same set of attributes.

   The xml lang: P

* History of Db Systems:

-> Punched cards invented by Herman Hollerith used in the beginning of 20th century to record U.S. census data & mechanical systems were used to process the cards & tabulate results.

-> 1950's & early 1960's: Magnetic tapes were developed for data storage. Processing of data consisted of reading data from one or more tapes and writes data into a ~~two~~ new tape.

-> late 1960's & 1990's: Widespread use of harddisks in the late 1960's changed the scenario for data processing; it allows direct access to data. Network & hierarchial db's could be created that allow data structures such as lists & trees to be stored on disk. Programmer could construct & manipulate these ds.

codd [1970] designed the relational model & non-procedural ways of Querying data in the relational model.

-> codd won the prestigious Association of computing machine Turing Award for his work.

Systemk, provided by Astrahan et all [1976] and Chamberlin et al [1981]

-> IBM's first relational db product are SQL/DS, IB DB2; Oracle, Ingres & DEC db. Therefore, relational db had replaced the n/w & hierarchial db & it becomes supreme among data models.

-> Early 1990's
primari
which a
processi
intensiv

Many
product
relationa

-> late 1990
DB had
dat rat
Db syste
to dat

=> Early
of XM
as a

- Early 1990's: The SQL lang. was designed primarily for decision support DS apps, which are query intensive & transaction processing app^n's, which are update intensive.

  Many db vendors introduced parallel db product in this period & also add obj-relational support to their dbs.

- Late 1990's: Explosive growth of the www. DB had to support high trans^n processing rates, high reliability & 24x7 availability. Db systems also had to support web interfaces to data.

- Early 2000's: In the early 2000's, the Emerging of XML & the associated Query lang. XQuery as a new db technology.

24.12.16

Introduction to DataBase Design:

The Entity Relationship data model allows us to describe the data involved in a real world Enterprise in terms of objects or things & their relationships.

-> The database design process can be divided into 6 types, the ER model is most relevant to the first three steps., remaining beyond E-R model

i) Requirement Analysis:- The very 1st step in desging the data base appn is to undrstand "what data is to be stored in the data base, what appn to be build on top of what Operation to be performed".

Several Methodologies have been proposed for organizing & presenting information gathered. The Outcome of this phase is a specification of user requirements.

ii) Conceptual database design:- The schema developed at this conceptual design phase provides a detailed overview of the Enterprise.

The Conceptual schema specifies the entities that are represented in the database, the attributes of the entities, the relationships b/w the Entities & constraints on the entities.

It provides a Graphical representation of the Schema.

iii) Logical DataBase Design: The designer maps the high level conceptual schema on to the implementation data model of the db system mapping the conceptual schema

using the
schema.
The logi
to avoid
iv) Schema
Relatio
schema
to refl
v) Physical
involv
cluster
of the
earlie
vi) Applic
proje
consid
Deri
Modelle
complete
cycle
for
db tha
-* E-R
Enti
-> An E
world
Ex:
-> An
type
attri
ex

using the E-R model in into a relational schema.

The logical database design is normalized to avoid redundancy.

iv) Schema Refinement: Analyze the collection of relations in our relational database schema to identify potential problems and to refine it.

v) Physical Database Design: This step may simply involve building indexes on some tables and clustering some tables or redesign of parts of the database schema obtained from the earlier design steps.

vi) Application & Security Design: Any software project that involves a dbms must consider aspects of application.

Design methodologies like UML (Unified Modelling Language) try to address the complete software design and development cycle.

for each role we must identify parts of db that must be accessible.

** E-R diagrams: Entities, Attributes and Entity sets.

→ An Entity is a "thing or Object" in the real world. i.e distinguishable from all other objects

Ex:- Each person in an enterprise is an Entity.

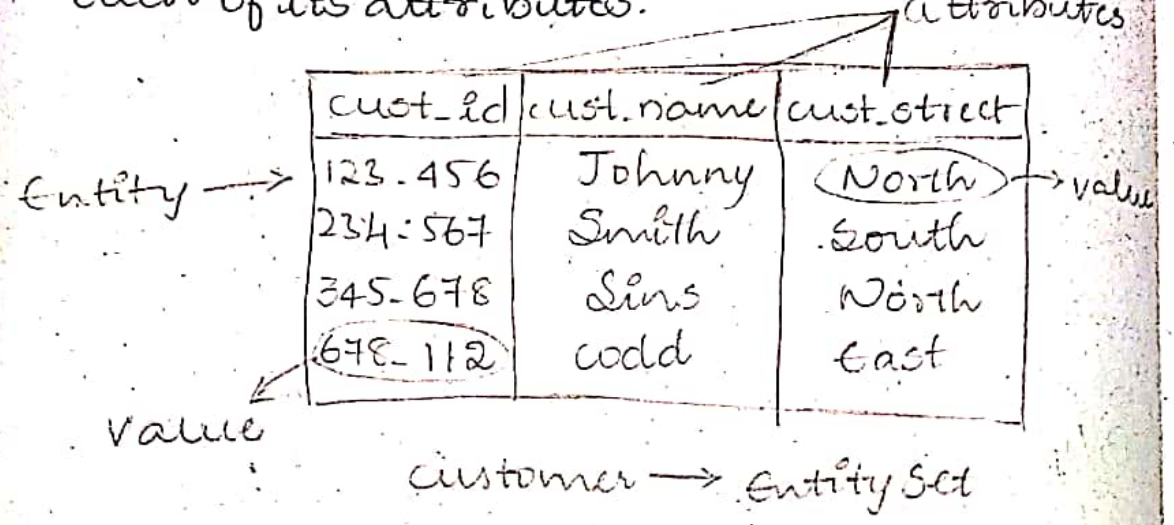
→ An Entity set is a set of Entities of the same type that share the same properties or attributes

Ex: Set of all persons who are customers at a

given bank, then customer is defined as an entity set

→ An entity is represented by a set of Attributes.

A set of Attributes values uniquely identify an Entity. Each entity has a value for each of its attributes.



→ for each Attribute associated with an entity set, Identify a domain of possible values.

Domain

Create table Customer (cust_id varchar(7), cust_name char(10), cust-street varchar(5))

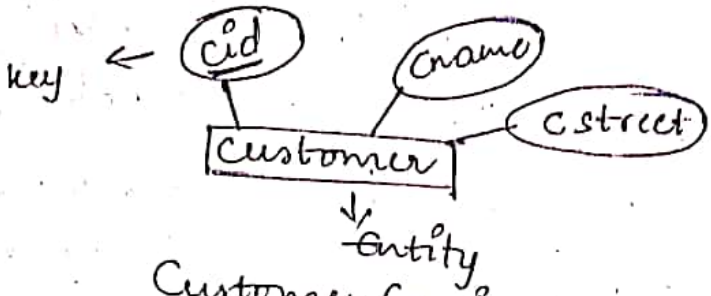
→ cust_id ranges from 1 to 7 supports 256 characters.

→ for each entity set there will be a key.

→ A key is a minimal set of attributes whose values uniquely identify an entity in a set.

→ There may be more than one key.

→ An entity set is represented by 'Rectangle'
→ An attribute is represented by 'oval'.
→ Each attribute in primary key is "underlined".



Customer Entity Set.

* Relationships and Relationships Sets:

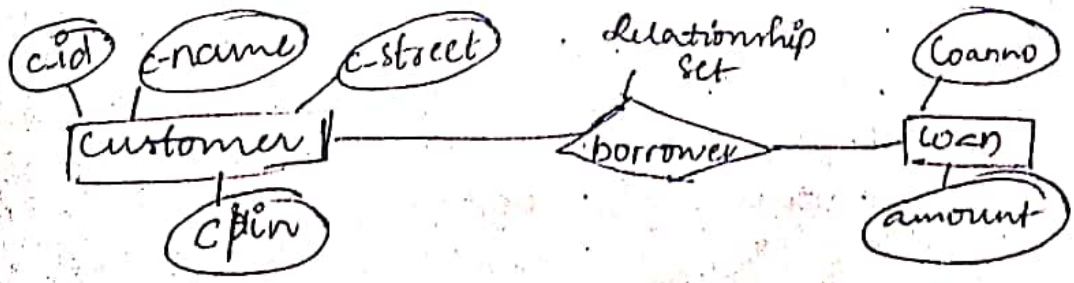
→ A relationship is an association among two or more Entities.

→ A relationship set is a set of relationships of the same type.

→ If $e_1, e_2, \ldots, e_n$ are entity sets, where $(e_1, e_2, \ldots e_n)$ is a relationship then the relationship set r is a subset of (where $n \geq 2$)

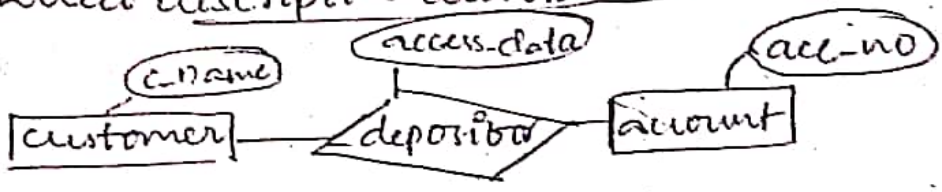
$\{(e_1, e_2, \ldots, e_n) / e_1 \in E_1, e_2 \in E_2, \ldots e_n \in E_n\}$.

→ Consider 2 Entity Sets customers & loan And the relationship set borrower to denote the association b/w customers and the bank loans that the customers have.



→ A relation is represented in the diagonal box or rhombus

→ A relationship may also have attributes called descriptive attributes.



→ Depositor is the relationship set b/w Entity sets customer & account.

→ Attribute access_data to the relationship is to specify the recent data on which a customer accessed an account

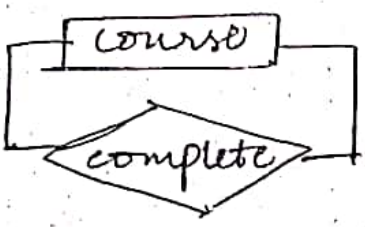
Eg:

Fig: Student class Relationship.

* Degree: The no. of Entities associated with a relationship set is known by degree of Entities.

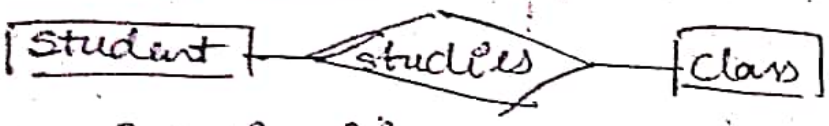
* Unary Relationship:

Where the association is within a single Entity

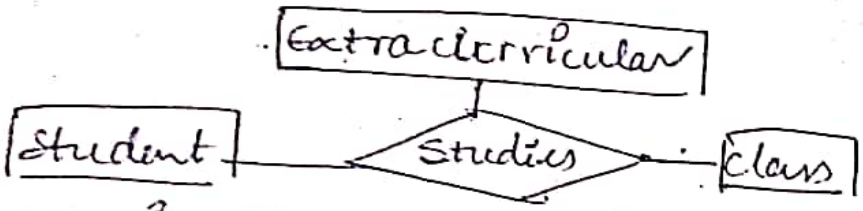


Eg: U cannot opt 'Java' unless the coourse in 'C' completes.

* Binary Relationship: A Relationship that associates two Entities is known as "binary Relationship".

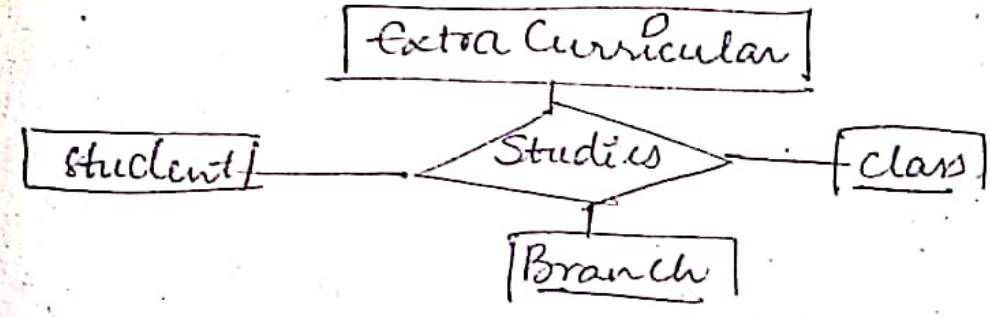


* Ternary Relationship:



A Relationship that associates 3 entities is known as "ternary Relationship"

* Quaternary Relationship:

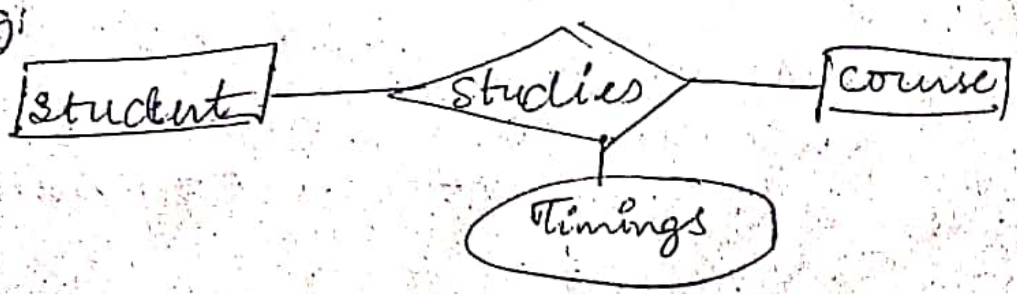


A Relationship that associates with 4 Entities is known as "Quaternary Relationship".

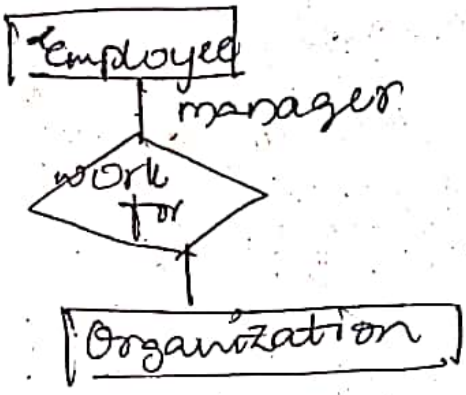
* Descriptive Attributes

(Attributes of Relationship)

Eg:

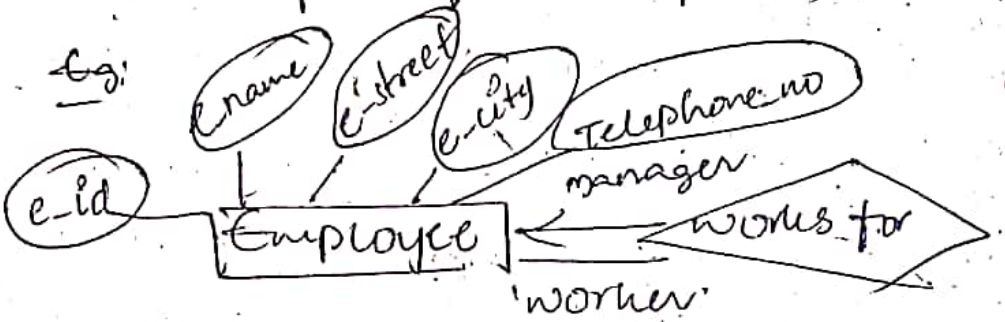


* Role: The function of the entity set in the relationship is called 'role'.

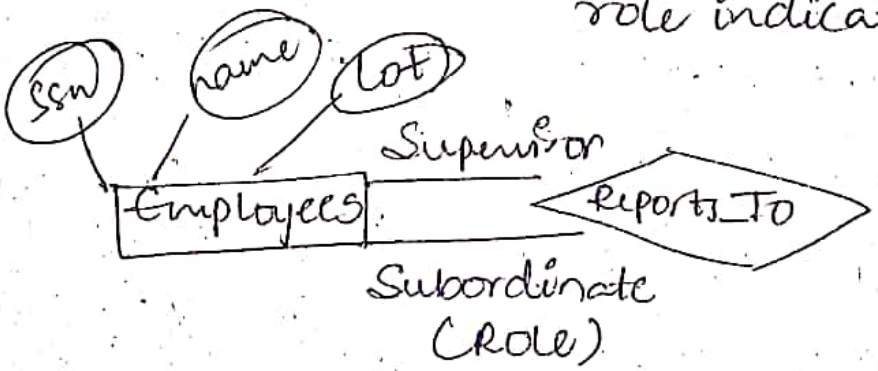


(To know the part of the employee)

Eg:



Eg. ER diagram with role indicators



* Types Of Attributes:

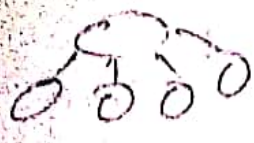
Each Attribute has a value set or data type associated with it.

i) Simple (Atomic) vs Composite Attributes:

Attributes that are not divisible are called Simple (or) Atomic Attributes

Eg: Age, id

An Attribute can be divided into smaller components is called Composite Attribute



Eg: Address , Name - First, last, Middle.
↓
Pincode, street etc.

ii) Single Valued Vs Multivalued Attribute

→ An Attribute having only one value is called as Single Valued Attribute.

Eg: D.O.B, Gender.

→ An Attribute having more than one value is called Multivalued Attribute

eg: Hobbies, phonenos

(Hobbies)

iii) Stored Attribute and Derived Attribute:

Attribute that cannot be derived from other Attribute Ex: D.O.B

The Attribute which are derived from another attribute or Multiple Attribute then that Attribute is known as Derived Attribute

Ex: age

(age)

iv) Null Value Attribute

A particular Entity may not have an applicable value for an attribute is specified as Null Value.

Ex: Degree

It doesn't specified in attribute, Spl

20.12.19

* Additional features of ER model:

Participation Constraints:

The participation of an entity set E in a relationship set R is set to be total if every entity E participates in atleast one relationship in R.

→ If only some entities in E participate in relationship in R, the participation of entity set 'E' in relationship 'R' is said to be partial.

Fig: Total participation of Entity set in a Relationship set

Weak Entity set:

- An Entity set do not possess its own primary key is known as weak Entity set, it is represented using a double rectangle '▭'.

Strong Entity Set: It posses its own primary key. It is represented using a Single rectangle '□'

for Relationship sets:

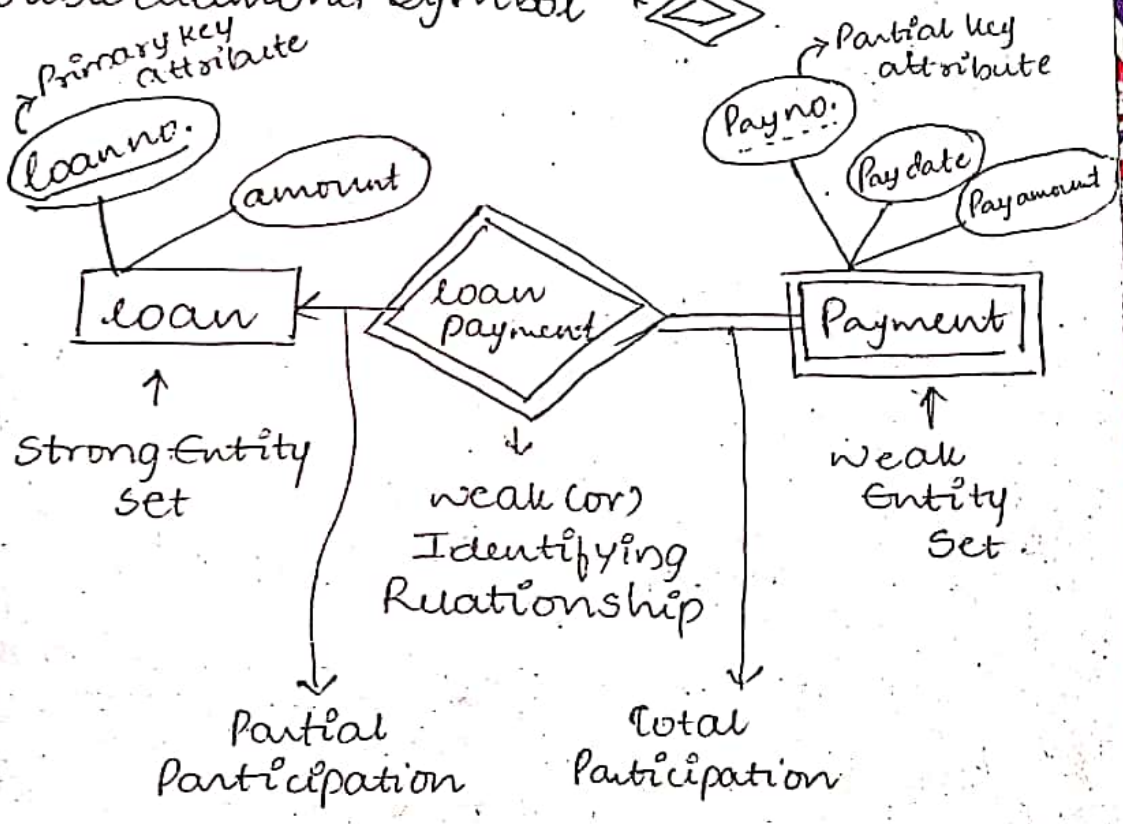
They are of two types

i) strong Relationship set

ii) weak Relationship set (or) Identifying Relationship set.

i) Strong Relationship set: It exist b/w two strong Entity sets. It is represented using a diamond symbol '◇'.

ii) Weak Relationship set: A weak or identifying relationship exist b/w the strong and weak entity set. It is represented using a double diamond symbol '◈'



→ It will be always many to one from the weak to strong Entity set & the participation of weak Entity set in the relationship is total.

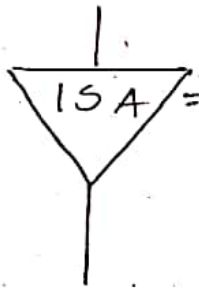
*Specialisation & Generalisation: (Class Hierarchi

Generalisation:

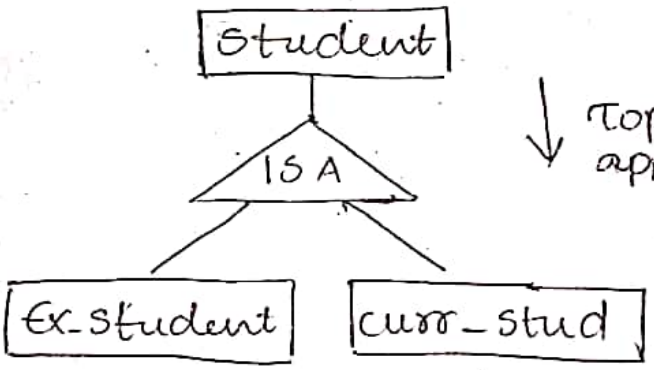
It is the process of forming a generalized superclass by extracting the common characteristics from two or more classes.

Specialisation:

It is a reverse process of Generalisation where a superclass is divided into subclasses by assigning the specific characteristics of subclasses to them.

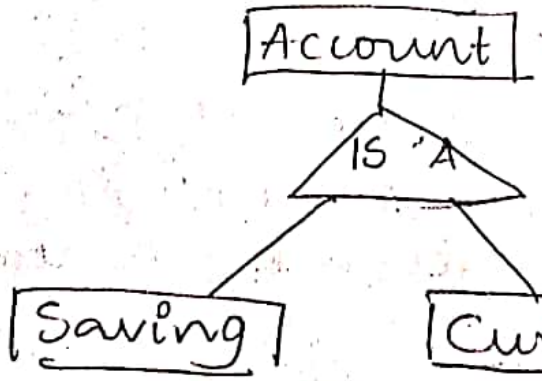


⇒ Is a specialization or Generalization.



↓ Top-down approach.

Fig:- Specialization.



↑ Bottom to top Approach

Fig: Generalization

Note: Overlap
present in c
superclass
constraints

If the de
to form a
constraint

*Cardinalit

It def
instances
There are
i) Many
one ins
with a
then it

for Ex:
Man
college
many c

Eg:

Note: Overlap constraints & Covering constraints present in class Hierarchies. If the details of superclass present in both the sub classes that constraints is known as Overlap constraints.

If the details of sub classes join together to form a base class is known as covering constraints.

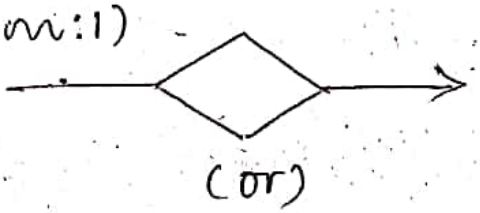
* Cardinality Constraints (or) Ratios:

It defines the max. no. of relationship instances in which an entity can participate. There are 4 types of Cardinality Ratios.

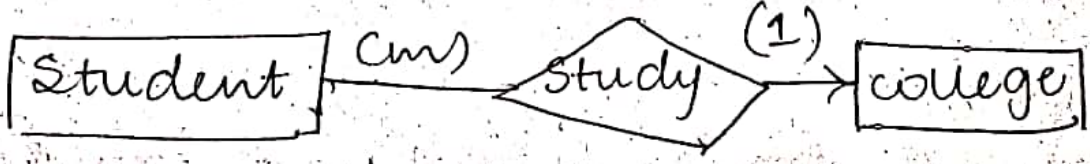
i) Many to One (M:1): when more than one instances of any entity is associated with a single instance of another entity then it is called Many to One Relationship.

for Ex:

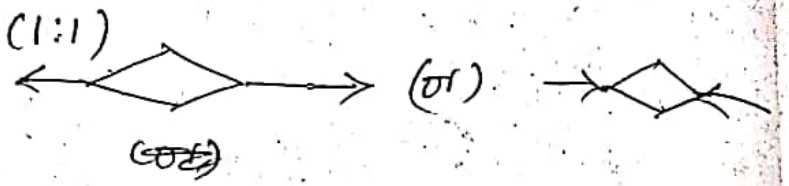
Many Students can study in a single college but a Student cannot study in many colleges at the same time.



Eg:



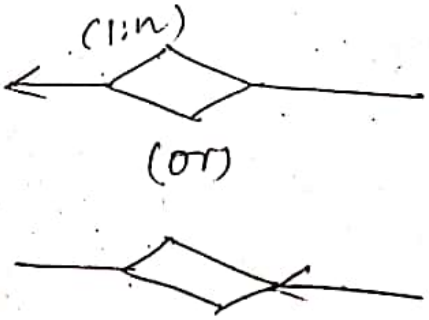
ii) One to One Relationship: When a single Instance of an Entity is associated with a single instance of another Entity then it is called One to One.



Ex: A person can have only one passport and a passoport is given to only one person.

iii) One to Many (1:N): when a single instance of an entity is associated with a more than one instance of another Entity, then it is called One to Many

Ex: A Customer can place many orders but a order cannot be placed by many customers



Eg:

iv) Many to

than one
associate
of anothe
to Many

Ex: A st
project
many s

st

* Applic

in wh
about
so, the
an One

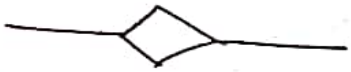
Eg:

In

Scanned by CamScanner

(iv) Many to Many Relationship: when more than one instances of an entity is associated with more than one instances of another Entity, then it is called Many to Many Relationship

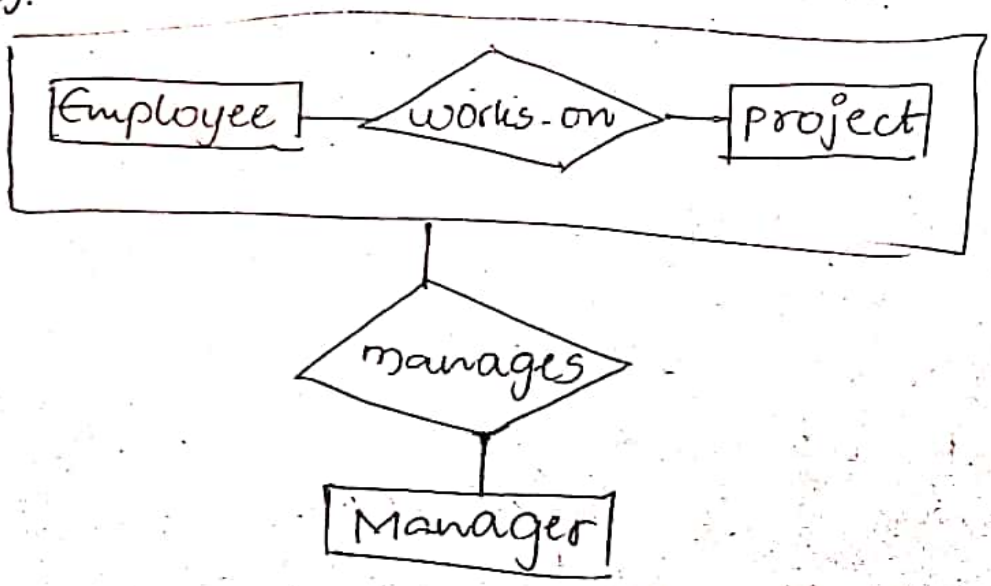
(m:n)



Ex: A student can be assigned to many project & a project can be assigned to many students

*Application: Aggregesion: It is a process in which a single Entity alone is not about to make sense in a relationship. So, the relationship of two Entities act as an One Entity

Eg:



In real world, A Manager not only manages the Employee working Under them

but he or she has to manage the project as well.

* Advantages & Disadvantages of E-R modelling:

Advantages:

i) E-R modelling is Simple & Easily Understandable
ii) Can help in data base design.
iii) Gives a higher level description of a system
iv) Can be generalized & specialized based on needs.
v) Intuitive and helps in physical data base creation.

Disadvantages:

i) physical design derived from E-R model may have some amount of ambiguities or inconsistencies.
ii) Sometimes diagrams may lead to misinterpretations.

* Conceptual Design with the E-R model:

Design choices:

i) Should a concept be modelled as an entity or an attribute?
ii) Should a concept be modelled as an entity or a relationship?
iii) Identifying relationships: Binary or Ternary? Aggregation?

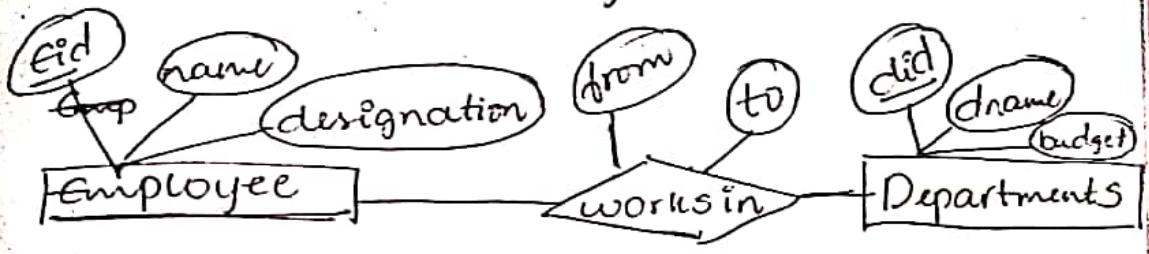
* Entity V/S Attribute:

Should Address be an attribute of an Employee or an Entity (connected to Employees by a relationship).

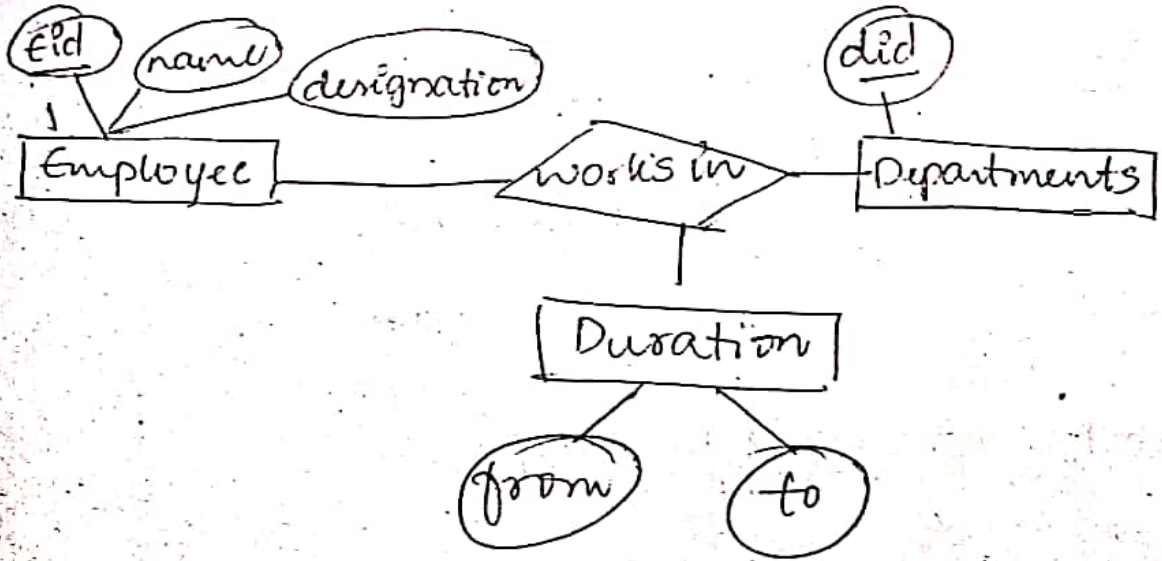
→ Depends upon the use we want to make of address information,

i) If we have several addresses per Employee address must be an Entity

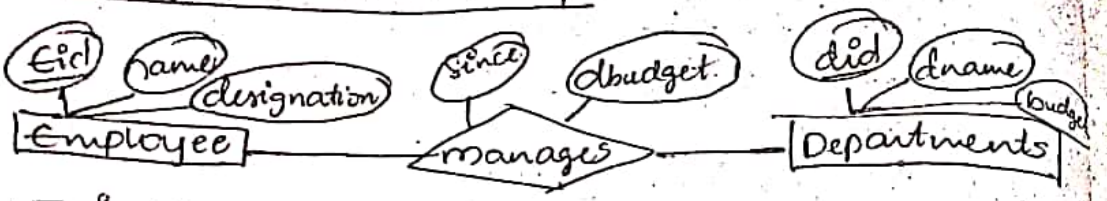
ii) If the structure (house no., street, city, State, ...) is important address must be modelled as an Entity.



If we want to record several values of the descriptive attributes for each instance of these relationship then we can address this problem by introducing an entity set called duration with attributes from and to.



Entity V/S Relationships:

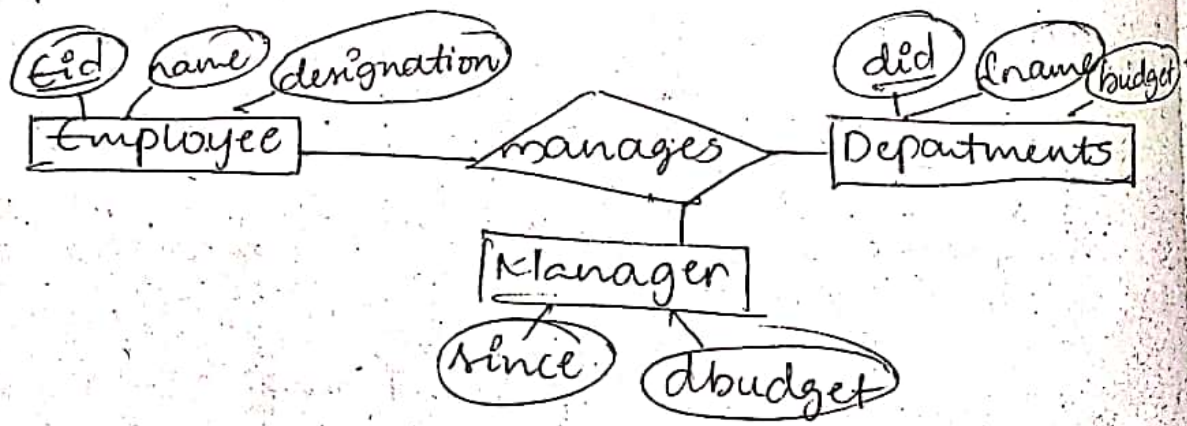


This ER diagram is Ok if a manages gets a separate discretionary budget for each dept. i.e dbudget,

But what if a manager gets a discretionary budget that covers all manage dept. This soln leads to redundancy of dbudget which is stored for each dept. managed by manager & it leads to misleading suggest dbudget tight to manage dept.

To Address this problem manager acts as an Entity set. The Attributes since & dbudget now describe a manager Entity.

-> In this case dbudget is an attribute of manger. But since is an attribute set b/w managers & departments



-> Binary V/S Ternary:

FROM TEXT BOOK

3/12/19

* Relational model:

Introduction to Relational model:

A RDBMS (Relational database) is a DBMS based on relational model introduced by E.F.codd.

In relational model data is stored in relations (tables) and is represented in the form of tuples (rows).

Relational data model is the primary data model which is used widely around the world for data storage and processing.

* Relational model concepts:

-> Table: A Table is a collection of data elements organised in terms of rows & column.
A Table is known as Relation

-> Tuple: A single Entry in a table is called a Tuple or rows or records
A Tuple in a table represents a set of related data.

-> Attributes: A Table consists of several records each record can be broken down into several smaller parts of data are known as Attributes (or) columns (or) field

-> Relation key: Each row has one or more attributes known as relation key which can identify the row in the relation or table uniquely.

-> Relation
relation
relati
Relati
tuples

-> Relatio
the re
and

-> Attrib
prede
dom

-> Degre
fiel
it is

-> card
no. o

Ex:

→ Relation Instance: A finite set of tuples in the relational database system represents relation instance.
Relation instance do not have duplicate tuples.

→ Relation Schema: A Relation Schema describes the relation name or table name, attributes and their names.

→ Attribute Domain: Each attribute has some predefined value scope known as attribute domain.

→ Degree of a Relation: It represents the no. of fields in a relation (Attributes). and it is also known as Arity

→ Cardinality of Relation: It represents the no. of tuples in an instance.

Ex: Table (or) Relation (or) file

Customer Fields or Attributes

Cid	Cname
1201	ABC
1202	XYZ
1203	XXX

→ Tuple or Record.

→ Degree (Arity) = 2
Cardinality = 3.

Integrity Constraints over Relations:

→ Data integrity refers to the validity and consistency of stored data. Integrity is usually Expressed in terms of constraints, which are consistency rules that the db is not permitted to violate.

→ Constraints may apply to each attribute or they may apply to relationships b/w tables.

→ Integrity Constraints Ensure that changes (Update deletion, insertion) made to the db by authorized users do not result in a loss of data consistency. Thus, integrity constraints guard against accidental damage to the db.

Example:

→ Blood group must be 'A' or 'B' or 'AB' or 'O' only (can not any other value also)

Types of Integrity Constraints:

Various types of Integrity Constraints are

1) Domain Integrity
2) Entity Integrity Constraint
3) Referential Integrity Constraint
4) Key Constraints.

1) Domain Integrity: It means the def^{n} of a valid set of values for an attribute. You define data type, length or size, is null value allowed, is the value unique or not for an attribute, the default value, the range (values in b/w) and or specific values for the attribute.

2) Entity Inte
This ru
value of attr
null.
eg: conside
a pk an
whereas o
values.

3) Referen
It s
in a re
watch a
or the t
The rule
1) It can
table i
table.
2) It c
primar
reco
3) It
the
the
4) It c
the pk
unre

2) Entity Integrity Constraints:

This rule states that in any db relation value of attribute of a primary key (PK) can't be null.

Eg: Consider a relation "STUDENT" where "id" is a pk and it must not contain any null value whereas other attributes may contain null values.

id	Name	Branch
1201	Ajay	CSE
1202	Suman	IT
1203	Barsha	
1204	Mortal	ECE
1205	Aman	CSE

3) Referential Integrity constant:

It states that if a foreign key (fk) exists in a relation then either the fk value must watch a pk value of some type in its relation or the fk value must be null.

The rules are:

1) U can't delete a record from a primary table if matching records exist in a related table.
2) U can't change a primary key value in the primary table if that record has related records.
3) U can't enter a value in the fk field of the related table that doesn't exist in the pk of the primary table.
4) However, U can enter a null value in the pk, specifying that the records are unrelated.

Eg: Consider 2 relation "student" and "student1" where 's_id' is the pk in the "student" relation and fk in the "student1" relation.

Student

sid	Name	Branch
1201	Ajay	CSE
1202	Suman	IT
1203	Asha	IT
1204	Aman	ECE
1205	Ram	CSE

Student 1

s_id	Course	Duration
1201	B.Tech	4 Yrs
1202	B.Tech	4 Yrs
1203	B.Tech	4 Yrs
1204	B.Tech	4 Yrs
1205	B.Tech	4 Yrs

4) key Constraints: → A key constraint is a statement that a certain minimal subset of the fields of a relation is a unique identifier for a tuple.

There are 4 types of key constraints:

i) Candidate key
ii) Super key
iii) Primary key
iv) foreign key

Types of Constraints:

1) NOT NULL
2) Unique
3) Check
4) key Constraints Pk, fk
5) Domain Constraints
6) Default

* Integrity Constraints / Enforcing Integrity Constraints:

→ NOT NULL - NOT NULL constraints makes sure that your column doesn't hold NULL value. When we don't provide value for a particular column while inserting a record into a table, it takes NULL value by default. But by specifying NOT NULL.

constraints, we
column(s) can

Eg:
create table

1) insert int
2) insert into
1st Stmt c
value insert
bcoz null

→ Unique -
column
values. 4
it mean
have du

Eg: create

3) Inser
4) Insert...
here,
3rd s
values
constra
4th
value

constraint, we have to sure that a particular column(s) cannot have NULL values.

eg:
```
create table student(rollno INT NOT NULL,
                     name varchar(20) NOT NULL,
                     age int NOT NULL,
                     address varchar(30));
```

1) insert into student values (101, 'Ram', 20, 'Hyd');
2) insert into student values(null, 'Asha', 20, 'Krmgr');

1st stmt can be executed bcoz no null value inserted, but the 2nd stmt is rejected bcoz null value inserted.

* unique - Unique constraint enforces a column or set of columns to have unique values. If a column has a unique constraint, it means that particular column cannot have duplicate values in a table.

eg:
```
create table student(rollno int NOT NULL unique,
                     name varchar(20) NOT NULL,
                     age int NOT NULL,
                     address varchar(30));
```

3) insert into student values (101, 'Arun', 20, null);
4) insert into student values (102, 'Rita', 20, null);

here,

3rd stmt cannot be inserted bcoz duplicate values in rollno. not accepted by unique constraint.

4th stmt can be inserted bcoz unique value has been inserted in rollno value.

→ Default: The default constraint provides a default value to a column when there is no value provided while inserting a record into a table.

-Eg:

```
create table student (rollno int unique NOT NULL,
                      name varchar(20) NOT NULL,
                      age int NOT NULL,
                      examfee int Default 10000,
                      address varchar(30));
```

5) insert into student values
(1203, 'Sita', 20, null, 'Rajahmundry');

6) insert into student values (1204, 'Ram', 21, 20000, 'Hyderabad');

Both stmts are inserted without any error

→ Check: This constraint is used for specifying range of values for a particular column for a table. When this constraint is being set on a column, it ensures that the specified column must have the value falling in the specified range.

Eg:

```
create table student (Roll no int UNIQUE,
                      name varchar(20) NOT NULL,
                      age int NOT NULL check (age>18),
                      address varchar(30));
```

Note: Every student age has to be >18 yrs.

7) Insert into student values (1205, 'Geeta', 17, 'Hyd');
8) Insert into student values (1206, 'Aman', null, 'Hyd');
9) Insert into student values (1205, 'Aman', 19, 'Hyd');

only 9th stmt. is inserted succesfully.

Key Constraints

Primary key: Uniquely identifies each record in a table. It must have unique values & cannot contains nulls.

Eg:

```
create table Student (Roll-no int,
                      name varchar(20) NOTNULL,
                      age int check (age>18),
                      Primary key (ROLL-NO));
```

In this example, roll-no field is marked as pk, that means the roll-no field cannot have duplicate & null values.

10) Insert into Student values (1206, 'Rizwan', 19);
11) Insert into Student values (1206, 'Lucky', 20);
12) Insert into Student values (null, 'Lusy', 20);

11th Stmt rejected due to duplicate value repeated in Roll_no. field.

12th Stmt rejected bcoz pk doesn't accept null values.

Foreign key (FK): FK's are the columns of a table that points to the pk of another table. They act as cross-reference b/w tables.

Domain Constraints:

Each table has certain set of columns & each column allows a same type of data, based on its datatype. The column doesnot accept values of any other data type.

Domain Constraint = Data type + Constraints.

Example of foreign key:

```
Create table student (sid char(10),        → Referenced relation
                      name char(20),
                      age int,
                      login varchar(35),
                      grade real,
                      primary key (sid));

Create table Enrolled (studid char(10),
                       cid char(15),
                       grade real,
                       foreignkey (studid)
                       references Student);
```

Referencing relation.

04:01:20

1) What should we do if an enrolled row is inserted, with a studid column value that does not appear in any row of the student table?

In this case, the insert command is simply rejected.

2) What should we do if a student row is deleted.

The Operations are

i) Delete all enrolled rows that refer to the deleted student row.

ii) Disallow the deletion of the student row if an enrolled row refers to it.

iii) Set the studid column to the sid of some (existing) 'default' student, for every enrolled row that refers to the deleted student row.

iv) For every e
the studid
(If studid
executed
oct).

3) What sh
of a stud
same as

SQL allo
options on

i) on Delete
ii) on Delete
iii) on Delet
iv) on Delet

Eg:
create t

* Query

→ A rel
about
new re
Query.
for wri
→ SQL i
Langu

iv) for every enrolled row that refers to it, set the studid column to null.
(If studid in enrolled is not pk then it is executed otherwise only i,ii,iii options are oct).

3) What should we do if the primary key value of a students row is updated?
Same as 2Ans Option.

SQL allows us to choose any of the four options on DELETE & UPDATE.

i) on Delete Cascade
ii) on Delete NO Action
iii) on Delete set Default
iv) on Delete set NULL

i) on UPDATE CASCADE
ii) on UPDATE REJECT
iii) on UPDATE SET DEFAULT
iv) on UPDATE SET NULL

Eg:
create table enrolled (studid char(10),
cid char(15),
grade char(2),
foreign key(studid) references
students
on delete cascade
on update NO ACTION)

* Querying Relational data:-

→ A relational db query is a question about the data & the answer consists of a new relation containing the result. A Query Language is a specialized language for writing Queries.

→ SQL is the most popular commercial Query language for a relational DBMS.

Eg: Select * from student where age > 18
↓
filter the records & fetching only necessary records.

→ 'where' clause is used to specify a condition while fetching the data from single table or joining with multiple table.

→ 'where' clause is also used in updata, delete, Stmt.

Syntax:

SELECT colname1, colname2, ... colname from tablename where [condition]
↓
using Comparision or logical Opeations like >, <, =, LIKE, NOT, AND, OR, BETWEEN etc.

* Displaying all columns:

Eg: select * from tablename;
select * from tablename where condition;

→ Displaying less columns:

Select column list from tablename;
select column list from tablename where condition;

→ Two Tables:

Select * from tablename1, tablename2 where
col1.tb1 = col1.tb2;

To sort the results of the Query by a given column (ORDER BY).

Select * from student ORDER by ROLLNO;
(By default in ascending Order)

Select * from student ORDER by ROLLNO desc;
(descending Order)

select * from stu

→ No Duplicate Ro
select DISTINCT (
select DISTINCT(

06.01.20

Logical Databa

→ ER diagram
relational m
→ This is beca
easily implem
Oracle etc.
following r
ER diagram
Rule 1: For S
Attribut
→ A strong e
will require
model.
→ Attribute
of the ent
→ The prim
the key
Eg: ROLL
St

Schema
Str

select * from student ORDER by ROLLNO. age;

→ No Duplicate Raws in Output:

select DISTINCT (name) from student;

select DISTINCT(*) from student;

06.01.20

Logical Database Design: ER to Relational.

-> ER diagram is converted into the tables in relational model.

-> This is because relational models can be easily implemented by RDBMS like MySQL, Oracle etc.

following rules are used for converting an ER diagram into the tables.

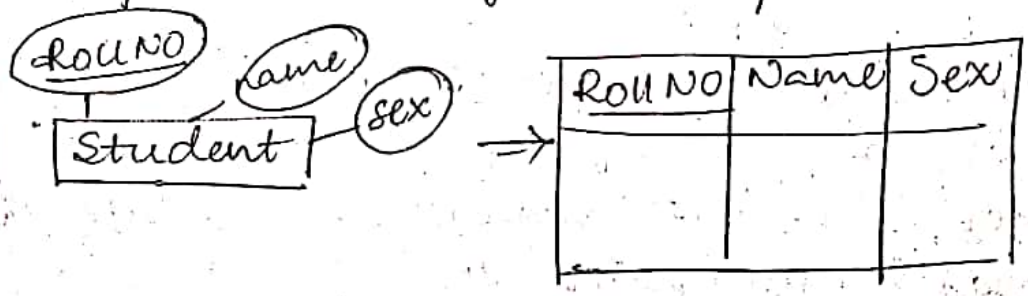
Rule 1: For strong entity set with only single Attribute.

→ A strong entity set with only simple attribute will require only one table in relational model.

-> Attributes of the table will be the attributes of the Entity set.

-> The primary key of the table will be the key attribute of the Entity set.

Eg:



Schema:

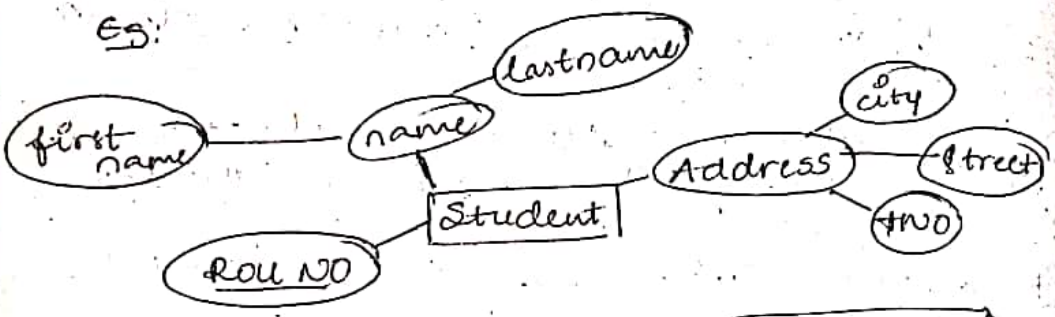
Student(ROLLNO, Name, Sex)

Rule 2: for strong Entity set with composition Attributes.

→ A strong Entity set with any no. of composition attributes will require only one table in relational model.

→ while conversion, simple attributes of the composite attributes are taken into account and not the composite attribute itself.

Eg:



ROLL NO	fname	lname	HNO	city	Street

Schema: Student (ROLL NO, fname, lname, HNO, street, city)

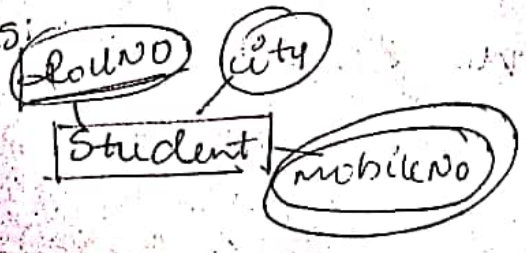
Rule 3: for strong entity set with multi valued attributes.

A strong entity set with any no. of multi valued attributes will require two tables in relational model.

→ One table will contain all the simple attribute with the primary key.

→ Other table will contain primary key & all the multi valued attributes

Eg:



RollNO	city

RollNO	Mobile

Rule 4: Translat

Table.

A relationship the relational

Attributes of the

→ pk attributes

→ Its own desc non-descript

Eg.

works in

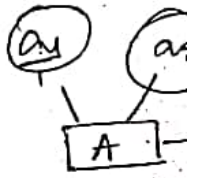
→ 3 Table

NOTE: one table entity set "E table for the "Department" one table fo

Rule 5: for ratios.

The follo

case 1: Bin



-Here, 3

1. A Ca

2. R C

3. B C

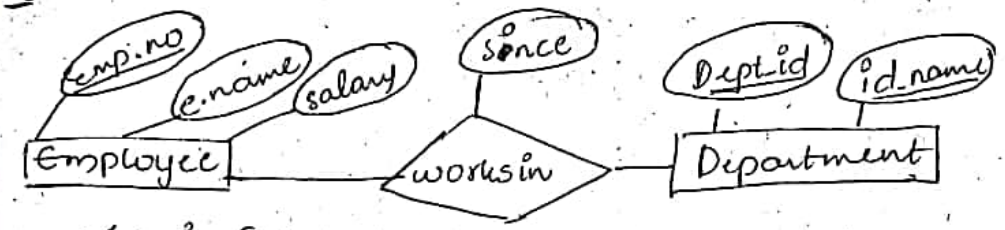
Rule 4: Translating relationship set into a Table.

A relationship set will require one table in the relational model.

Attributes of the table are:

→ pk attributes of the participating entity sets.
→ Its own descriptive attributes if any set of non-descriptive will be the pk.

Eg.



works in (Emp_no, Dept_id, since)

→ 3 Table required in ~~the~~ relational model

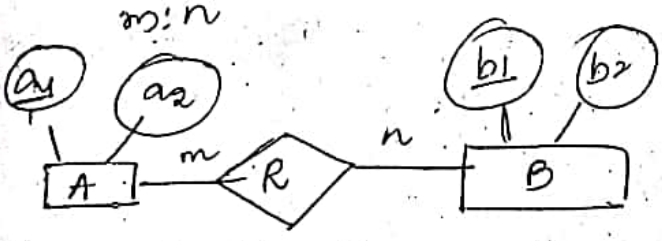
NOTE: one table for the entity set "Employee" one table for the entity set "Department".
One table for the relationship ~~set~~ set works in.

Emp.no.	Dept. id	since

Rule 5: For Binary relationships with cardinality ratios.

The following four cases are possible.

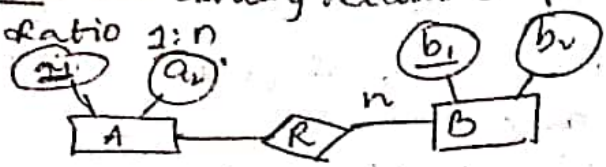
Case 1: Binary relationship with Cardinality Ratio m:n



Here, 3 tables Required:

1. A(a1, a2).
2. R(a1, b1)
3. B(b1, b2)

Case 2: for Binary relationship with Cardinality ratio 1:n

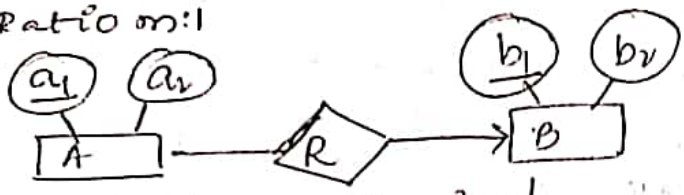


Here, 2 tables Required.

1. $A(\underline{a_1}, a_2)$
2. $BR(a_1, \underline{b_1}, b_2)$

NOTE: Combined ~~state~~ table will be drawn from the entity set B & relationship set R.

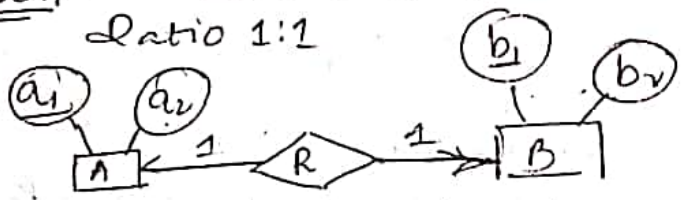
Case 3: for Binary relationship with Cardinality ratio m:1



Here, 2 tables are required.

1. $AR(\underline{a_1}, a_2, b_1)$ → NOTE: Combined table will be drawn for the entity set A & Relationship set R.
2. $B(\underline{b_1}, b_2)$

case 4: for Relationship set with Cardinality ratio 1:1



Here, two tables will be required.

Either combine 'R' with 'A' or 'B'

way 1: 1. $AR(\underline{a_1}, a_2, b_1)$
2. $B(\underline{b_1}, b_2)$

way 2: 1. $A(\underline{a_1}, a_2)$
2. $BR(a_1, \underline{b_1}, b_2)$

Rule 6: for cardinality constraints
→ Cardinali discuss se
→ Because foreign i.e non

case 1:
constra one sid

Then, tu

Bcoz, acquir null

case 2
const from
If t side then usi

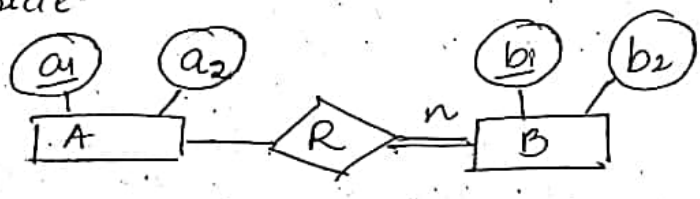
Here

Rule 6: for Binary relationships with both cardinality constraints & participation constraints.

→ Cardinality constraints will be implemented as discussed in Rule 5.

→ Because of the total participation constraint, foreign key acquires NOT NULL constraint i.e now fk cannot be null.

case 1: for primary relationship with cardinality constraint & total participation constraint from one side.

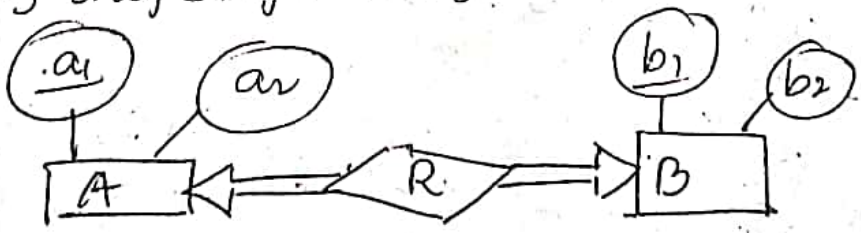


Then, two tables required - 1. A ($\underline{a_1}$, a_2)
2. BE (a_1, $\underline{b_1}$, b_2)

Bcoz, of total participation fk all has acquired NOT NULL constraint, so it can't be null now.

case 2: for Binary relationship with cardinality constraint & Total participation constraint from both sides.

If there is a key Constraint from both the sides of an Entity set with Total participation, then that binary relationship is represented using only single table.

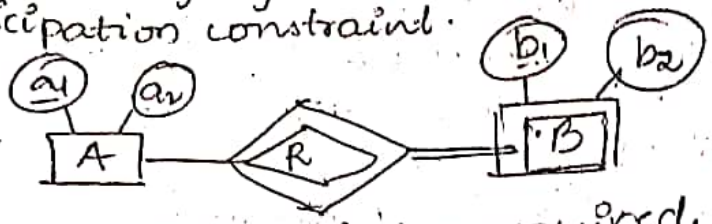


Here, only one table is required,

ARB ($\underline{a_1}$, a_2, $\underline{b_1}$, b_2)

Rule 7: For Binary-Relationship with weak entity set.

Weak entity set always appears in association with identifying relation with total participation constraint.



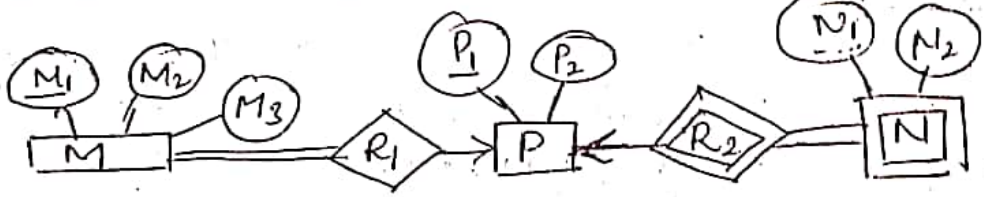
Here, two table will be required,

1. A ($\underline{a_1}$, a_2)
2. BR ($\underline{a_1}$, $\underline{b_1}$, b_2)

07.01.20.

Examples: ER Diagrams in Relational Model.

1)



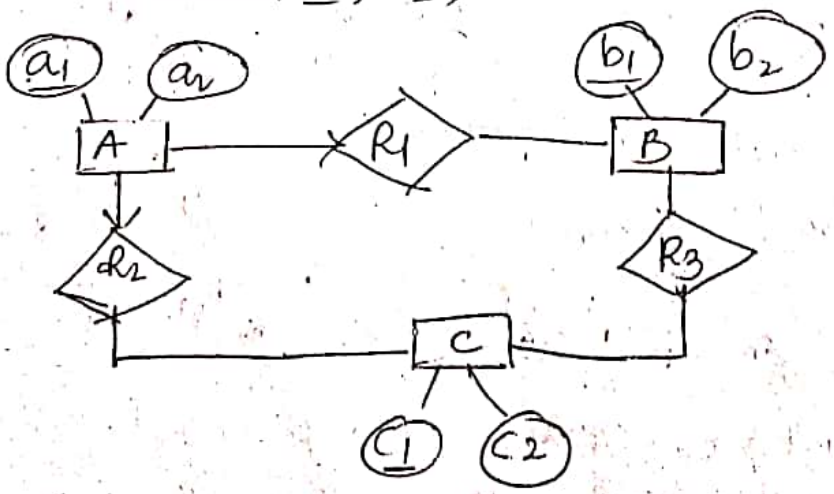
D: Applying the rules, min 3 tables will be required.

MR1 (M1, M2, M3, P1)

P (P1, P2)

NR2 (P1, N1, N2)

2)



3) Applying the rules, min. 4 tables will be required.

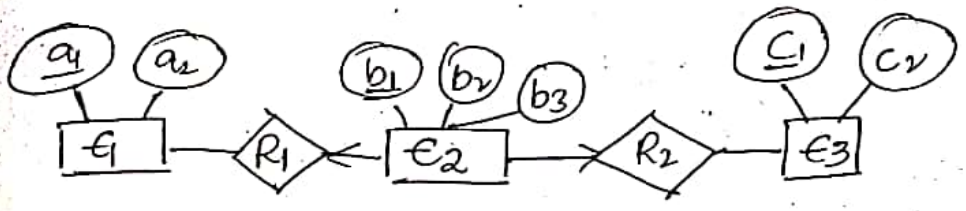
A $R_1 R_2 (\underline{a_1}, a_2, \underline{b_1}, \underline{c_1})$

B $(\underline{b_1}, b_2)$

C $(\underline{c_1}, c_2)$

$R_3 (\underline{b_1}, \underline{c_1})$

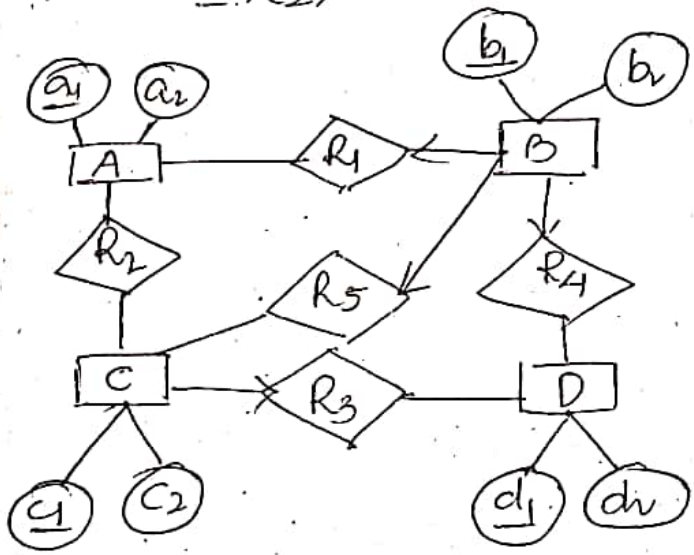
3)



2)

- $E_1 (\underline{a_1}, a_2)$
- $E_2 R_1 R_2 (\underline{b_1}, b_2, b_3, \underline{a_1}, \underline{c_1})$
- $E_3 (\underline{c_1}, c_2)$

4)



A:

$B R_1 R_4 R_5 (\underline{b_1}, b_2, \underline{a_1}, \underline{d_1}, \underline{c_1})$

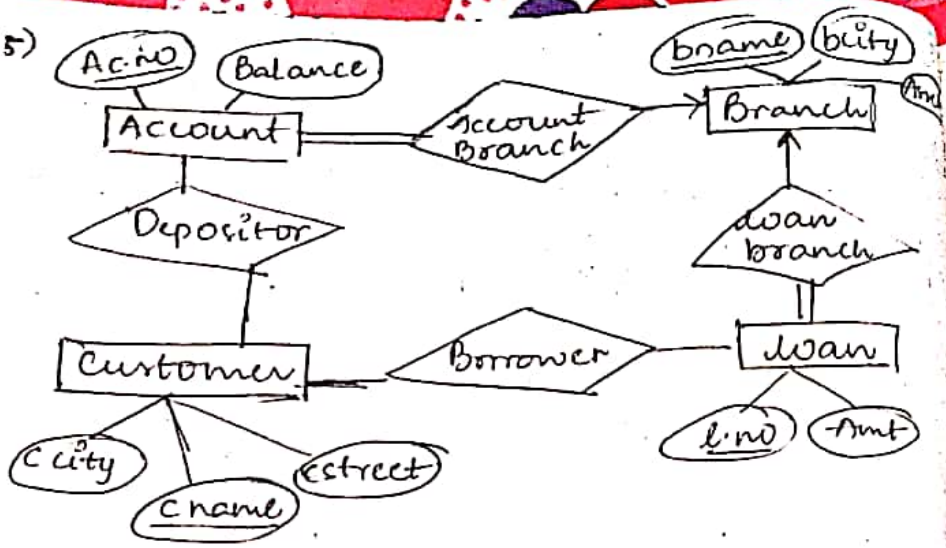
$A (\underline{a_1}, a_2)$

$C R_3 (\underline{c_1}, c_2, \underline{d_1})$

$d (\underline{d_1}, d_2)$

$R_2 (\underline{a_1}, \underline{c_1})$

5)



A) - Account (AC-NO, Balance, b_name)
Branch (b_name, b_city, Assets)
loan (l_no, Amt, b_name)
Borrower (C_name, l_no)
~~Cccc~~ Customer (C_name, C_street, C_city)
Depositor (AC-NO, C_name)

* Introduction to Views:

Views in SQL are kind of virtual tables. A view also has rows & columns as they are in a real table in the db. We can create a view by selecting fields from one or more tables present in the db. A view can either have all the rows of a table or specific rows based on certain condition

Student details

S_id	name	Address
1	Harish	Hyd
2.	Harsha	Bihar
3.	Pratik	Delhi
4.	Ram	Delhi
5.	Ramesh	Hyd

Student Marks

id	name	marks	Age
1.	Harish	90	19
2.	Harsha	50	20
3.	pratik	80	19
4.	Ram	95	21
5.	Ramesh	85	18

A view can be cr
tables. We can cre
VIEW statement.

Syntax: create
select
from
whe

view-name: N
table-name:
Condition :

Eg 1) Creating

create view
select name,
from stude
where S_I

Select

O/P:

Eg2) Create
Select
from
Order

Select *

A view can be created from a single or multiple tables. We can create view using ~~create view~~ CREATE VIEW statement.

Syntax: Create View View_name AS
Select col1, col2, ...
from table_name
where condition;

view_name: Name for the view
table_name: Name of the table(s)
Condition : Condition to select rows.

Eg 1) Creating view from a single table:

create view Detailsview AS
select name, address
from Studentdetails
where S_ID < 5;

O/P: View created.

Select * from Detailsview;

O/P:

Name	Address
Harish	Hyd
Harsha	Bihar
Pratik	Delhi
Ram	Delhi

Eg2) Create view student as
Select id, name
from Studentmarks
Order by age;

Select * from student;

=>

id	name
5	Ramesh
1	Harish
3	Pratik
2	Harsha
4	Ram

Creating view from multiple tables:

Eg

1) Create view marksview AS
select studentdetails.name, studentdetails.address, studentmarks.marks from studentdetails, studentmarks where studentdetails.s_id = studentmarks.id;
select * from marksview;

Name	Address	marks
Harish	Hyd	90
Hasha	Bihar	50
Ratik	Delhi	80
Ram	Delhi	95
Ramah	Hyd	85

deleting views:

Syntax: drop view viewname; (Name of the view which we want to delete)

Eg: drop view marksview;

Eg: Create view Students (name, id, score)
as select s.name, s-id, m. marks
from studentdetails s, Studentmarks m
where s. s-id = m.id;

* Destroying/Altering Tables and Views:

Updating views: There are certain conditions needed to be satisfied to update a view

1) The select stmt which is used to create the view should not include GROUPBY clause or ORDERBY clause.

2) The SELECT keyword.
3) The view s
4) The view sh queries or
5) The view If the vie then we u view.

2) The SELECT Stmt should not have the DISTINCT keyword.
3) The view should have all NOTNULL values
4) The view should not be created using nested queries or Complex Queries
5) The view should be created from single table. If the view is created using multiple tables then we will not be allowed to update the view.