Sumedh

- Topics

1. File processing System
2. DBMS
3. Architecture
4. Data Independance
5. Evolution of Models
6. ER Model
7. Relational Model

- File Processing System

A file processing is like a basic db system that lets u work with one file | table at a time

data is stored directly in files without linking

- Issues

1. data rebundancy

Problem : Same data stored in diff files

Impact : hard maintainance
Wask of storage

2. data Inconsistency

Problem : Change in one file aren't reflected in others

Impact : decisions on wrong data
No accurate data

3. Ltd data sharing

Problem : files hard to share blw systems

Impact : Collab suffers
Outdated info

4. data denpendancy

Problem : progs tied to specific file format

Impact : High maintainace
any change req reworking of progr

5. Lack of data Integrity

Problem : No buit-in checks for correct data

Impact : more erras
poor data quality

6. Ltd security problem

Problem : Weak protection against unauth access

Impact : data breach
Security compromised

7. Concurrency Control

Problem : difficlt to manage multiple users

Impact : overwriting
data corruption

8. Scalability

Problem : struggle to mange growing data

Impact : slow performance
scalling challenges

9. Ltd query capabilities

Problem : No tools to search & analyse data
Impact : Hard to retrive
gen reports takes time

- DBMS Terminologies

1. Database - Collection of organised data that can be easily accessed, manged & updated

2. dbms - Software that helps users interact with db

3. Table - Organises data into row & colums

4. Row / Records - A single entry in a table

5. Column / field - Vertical section containing attributes

6. Primary key - A uniqe indentities for each row in

7. Foreign key - A column linking one table to another

8. Qury - A req to retrieve or manipulate data

9. Indux - A structure that makes data retrieval faster

10. Normalisation - Organising data to remove duplicates & improves accuracy

11. Relational db - Uses tables to organise data & define relationship

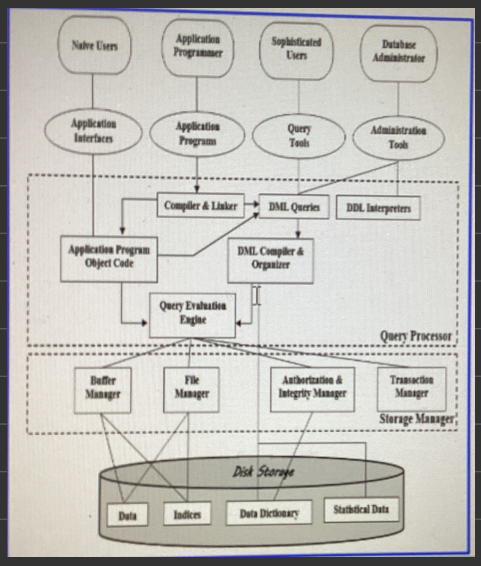
12. Transaction - A grp of ops as one unit to maintain consistency

13. ACID - Atomicity : all of nothing
Consistency : data stays valid
Isolations : tranactn don't interfere
Durability : permanent change

14 Schema - blueprint of db

15 data dictionary - Stores Metadata

- dbms architecture



- Types of db users

1. Naive Users - directly use db thru app
2. App programr - write program that interact with db
3. Sophisticated users - use adv tools / queries to analyse
4. db administrators - Manage & maintain db

- Query processorr

Translates queries into instruct^{n} for db

Key Components :-

1. DML Compiler - Converts data manipulation statemnts to low-lvl instruct^{n}

2. DDL Interpretor - Converts data definition statements into metadata

3 Embedded DML pre Compiler - processes DML statements in app progs

4. Query Optimizer - Executes & optimizes instructions for efficiency

- **Storage Manager**
  - Acts as an Interface blw db & user req
    Also ensures Consistency & integrity

Components :-

1. Authorization Mangr - Verifies user permissions
2. Integrity Manager - Make sure data follows integrity rules
3. Transaction Manager - Manages multiple user ops to maintain consistency
4. File Manger - Handles storage & data structure
5. Buffer Manager - Handles cache & temp secondary storage
6. Disk Storage - Stores all db into
   - i) files
   - ii) dictionary
   - iii) Indices (faster data retrieval)
   - iv) Statistical data

- **Data Independance**

It allows change in one db lvl w/o affecting others

- lvl of db

1. Physical Level (low-lvl) : disk storage, data access methods
2. Conceptual Level : Query logic, procedures & db struc
3. Logical lvl (view lvl) : User Interface & views

- Types of db Indep

1. Physical

Changes in physical storage don't affect logical structure

eg Changing storage devices
Modifying indexes / data s
Migrating db to new storag

2. Logical data Independance

Changes in the logical structure doesn't affect physical structure

eg adding, modifying / deleting attributes
merging splitting records

- Importance

1. Improves data quality & performance
2. Simplifies maintainance of dbms
3. Enhances db security
4. devlopers don't need to focus on internal structure

Difference Between Physical and Logical Data Independence

Aspect	Physical Data Independence	Logical Data Independence
Focus	Storage methods and devices	Data structure and definition
Ease of Retrieval	Easy	More complex due to logical dependency
Ease of Achievement	Easier	Harder
Concerned Schema	Physical Schema	Logical Schema

- Evolution of data Models

In order to prevent file system limitations, devlopers created data models to organise data

- Types

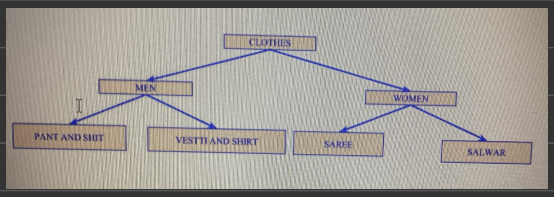
1. Hierarchical
2. Network
3. ER
4. Relational
5. Object - Oriented

• Hierarchical Model

• Into :-

1. first model used in dbms
2. data is organised in a tree-like structure
3. Easy to understand

• eg :



• Key features :-

1. One-to-many relat^nship
2. parent-child "
3. pointers
4. deletion problem

• Advantages :-

1. Simple & fast to navigate
2. Change in parent node automatically reflects in child

• disadv :-

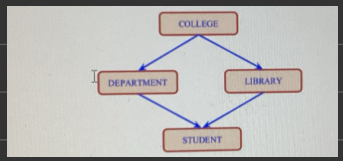
1. Complex in large db's
2. If parent deleted, child lost

• Network Model

• Into :-

1. Extension of hierarchical model
2. Child can have more than 1 parent

• eg :



- Key features :-

1. More relat^n ships
2. More paths
3. Circular LL

- Adv :-

1. faster data access
2. Change in parent reflect in child

- disadv :-

- More Complex due to multiple relationships

- ER Model

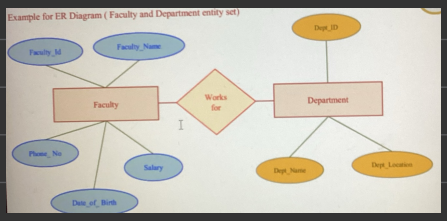
- Info

1. High-lvl data model
2. Represent real-world problems with pictorial diagram
3. easy for devlopers to understand

- Components :-

  - Entities
  - Attributes
  - Relationships

- eg :



- adv :-

1. Simple n easy to understand
2. helps in bettr Comm

- disadv :-

1. No Industry std
2. Some data may be hidden

- Relational Model

Info :-

1. The most widely used model
2. data is stored in tables

eg :

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800	-	20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975	-	20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	-	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	-	10
7788	SCOTT	ANALYST	7566	09-DEC-82	3000	-	20
7839	KING	PRESIDENT	-	17-NOV-81	5000	-	10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100	-	20
7900	JAMES	CLERK	7698	03-DEC-81	950	-	30
7902	FORD	ANALYST	7566	03-DEC-81	3000	-	20
7934	MILLER	CLERK	7782	23-JAN-82	1300	-	10

- Key features :-

1. Row (Record / tuple)
2. Column (Attribute / field)
3. Table (Entity set)

- adv :-

1 Simple & unscattered
2. Scalable and easy to manage

- disadv :-

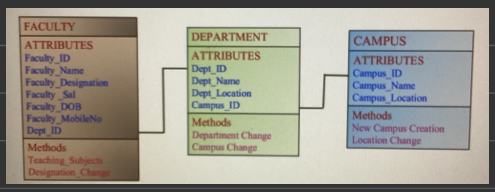
1. Requires more h/w resources

- Object - Oriented Model

Info :-

1. Represents real-life word problems using objects
2. data & relationship stored in single structure
3. Can store complex data like img, vids, audios
4 Obj connected thru links using attributes.

eg :



- Data Abstraction

  L db design starts with a high-lvl view & gets more detailed as it moves toward implementation

In 1970, ANSI / SPARC created a framework with 4 lvls of abstract^n

L
1. External
2. Conceptual
3. Internal
4. Physical

- External Mode :-

It's the end users view of the data, tailored to their specific needs

eg. Sales team see customer data

- Conceptual Model

The dbms view of db, showing tables, relationship & constraints

eg : A Customer table linked to orders table

- Internal Model :-

The technical view of how data stored & accessed, goal is to achieve logical independance

eg : Changing data indexing w/o altering user interact^n

- Physical Model :-

The lowest lvl dealing with hlw & storage details

eg : Storing data on a local server

• Entity - Relationship (E-R) Model

1. High-lvl data model
2. Represent real-world problems with pictorial diagram
3. easy for devlopers to understand
4. Require no tech knowledge
5. No hlw support needed

· Symbols used

1. Rectangles : represent entities
2. Ellipses : represent attributes
3. Lines : Connect attributes to entities & entity sets with other relation types
4. double ellipse : represent multi-valued attributes
5. double rect : represent weak entity

· Entity & entity sets

· Entity : An obj with physical / conceptual existence
eg person, job, etc

· Entity Set : set of all entities is a entity set
eg Students

Two types

└ i) Strong entity : type of entity with a key attribute, does not depend on any other entity
It has a uniqme primary key
(represented by rect)

ii) Weak entity : Here, key attributes cannot be defined, they're called weak entity type

eg parents, child, wife of a entity employee

dependant

(represented by double rect

- Attributes

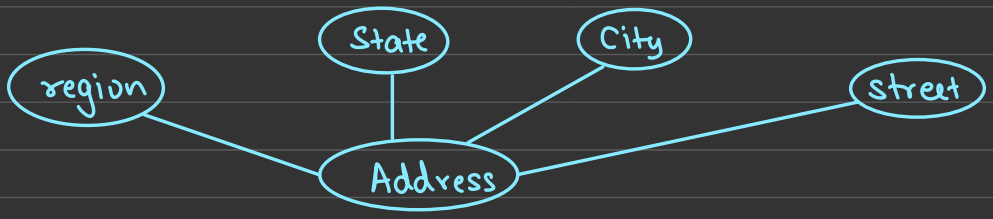
  L They are properties that define the entity type
  eg roll_No, DOB, Add, etc

  (Represented by ellipse)

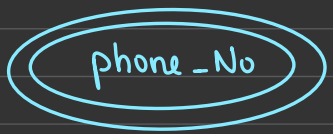
Types

1. Key attribute : Uniquely identifies each entity in the entity set
   eg roll_No

2. Composite attribute : It's composed of many other attributes
   eg address with street, city, region, etc

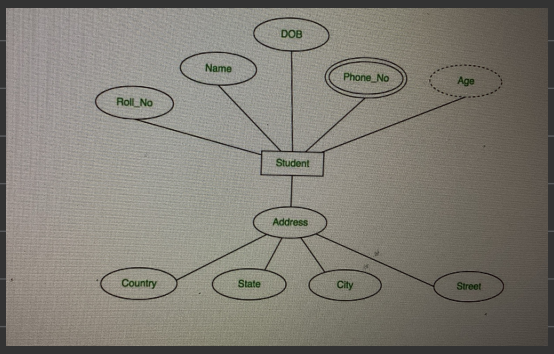


3. Multi-valued attribute : has more than one value for a given entity
   eg phone_No



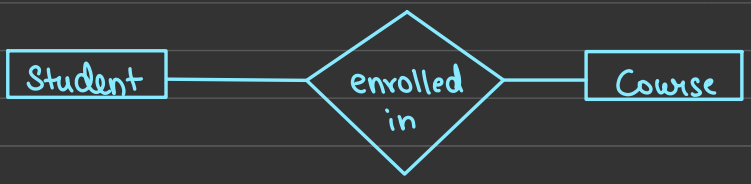
4. derived attribute : An att. that can be derived from others
eg age from DOB

All combined

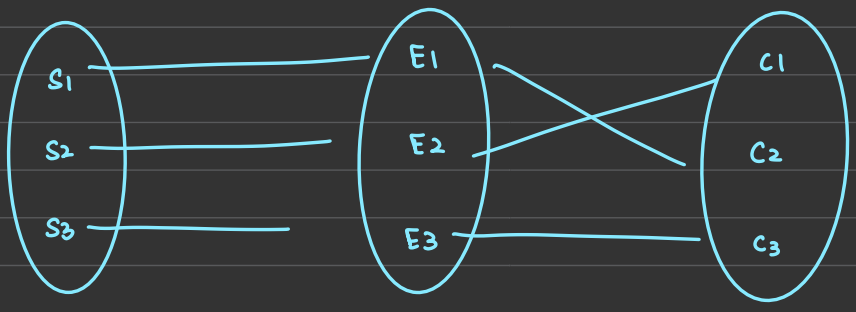


- Relationship & sets

Relationship : It represents the association blw entity types
(represented by diamond)



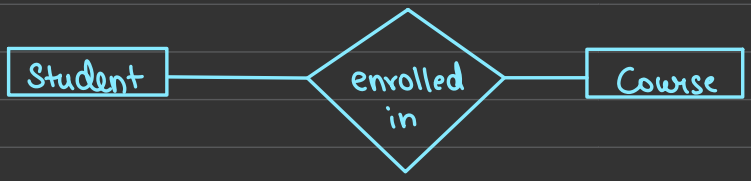
Entity-relationship set : A set of relationships of the same type is known as a relationship set



degrees in relat^{n}ship set

- **Unary relationship** : when only 1 entity set participating in a relation
  eg one person married to only one person

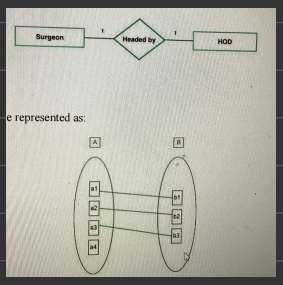
- **Binary relationship** : Two entities participating in a relation



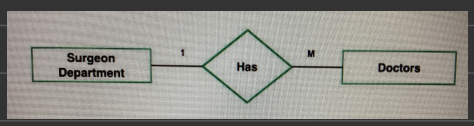
- **3-nary relationship** : n entities participating

Cardinalities

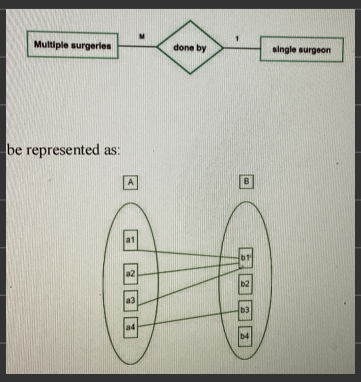
- **One - to One** : when each entity in a entity set take part only once



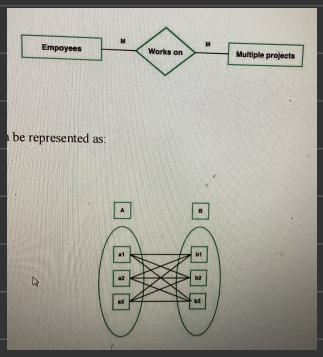
- **One - to Many** : each entity can be related to more than one relationship



- **Many-to-one** : when entities in one set can take part only once while entities of other can take more than once

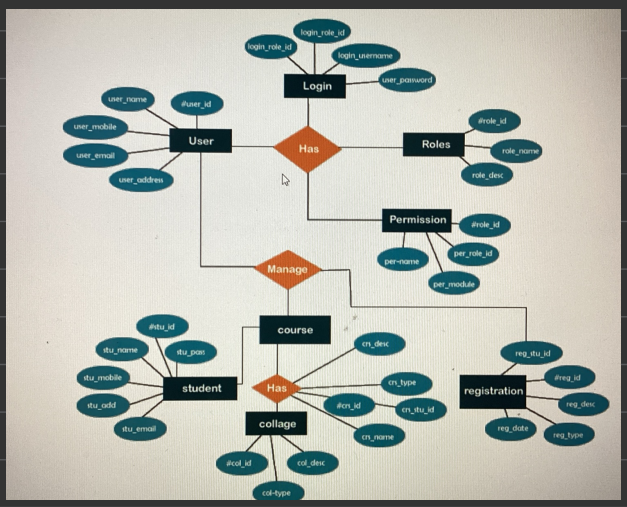


- **Many-to-Many** : "do whatever u want"

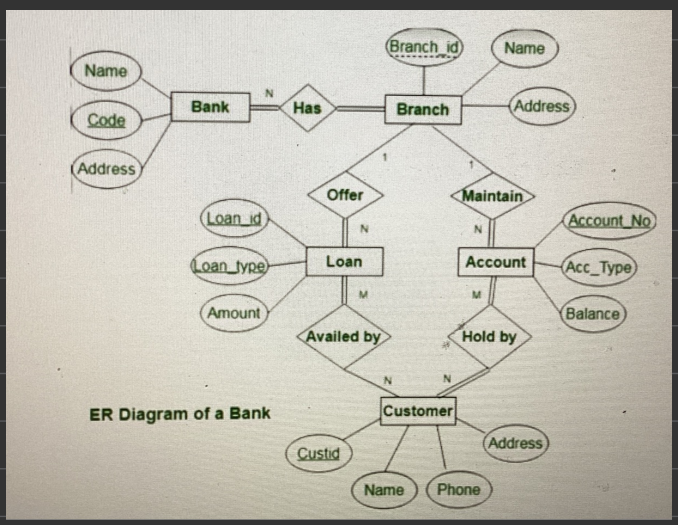


- **Examples :-**

University ER model



Banking system



- Extented ER features

Basic ER models handle most db needs, but extented ER features are used for more complex features

- i) Specilization
- ii) Generalization
- iii) Higher & lower entity sets
- iv) Aggregation

- Specialization :-

dividing an entity set into smaller subgyps based on unique attributes

eg : i) Students can be specilized to UG & PG

UG has hostel attribute
PG has office attribute

ii) Employees into faculty & secretary

faculty with designation attribute
secretary with hours_per_week "

- Generalisation

Combining multiple entity sets into a higher-lvl entity set based on common attributes

eg faculty & secretary has common attributes like id, name, etc. They can be generalised in a common entity set

- Aggregation :-

Treating a relationship as a higher-lvl entity

eg A 'project_guide' relationship blw 'faculty', 'student', 'project' can be treated as an entity to store additional info like review_report

• ER Issues

Designing an ER diagram involves making decisions about

1. Entity set v/s Attributes
2. " " v/s relationship sets
3. Binary v/s n-ary " "
4. Placement of relationship attributes

• Entity set v/s attributes

Issue : Should a property be an attribute or a seperate entity

eg : phone_no attribute

If phone_no is treated as an attribute the each faculty can have one phone_no n if entity then multiple phone_no

Solution : use entity set for props that req, more details

• Entity set v/s relationship sets

Issue : Should a " treated as rel. set or entity set

eg : Takes relation blw student & section can be replaced with reg_no

Solution : use entity set where relationship itself has additional attributes

• Binary v/s n-ary

Issue : Should a relship involve two or n entities

eg : parent (blw child, mother, father) can be replaced by mother (blw child & mother) & father (blw child & father)

Solution : Use binary as simple & more flexible

- Placement of relationship attributes

Issue : where shld attr. of a rel. placed

eg : In a Counselor rel (0 to M), the date att can be place on student

Solution : Place rel att in entity sets with (0 to 0) | (M to M) rel.

- Relational Model

- It's the most common db today
- Simple, making it easy for devlopers to use
- Replaced older models like network & hierarchical

- Struture

Here, data is stored in tables (relations) with row (tuples) & columns (attributes)

eg A faculty table with attr. like f_id, f_name, etc

- Terms in structure

1. relation = table
2. Tuple = row
3. attribute = Column
4. domain = set of valid values for attr.
5. Null val = unknown / missing value

- db schema

- Schema : logical struc of db
- Instance : data snapshot at a given time
- Schema includes rel & att.

eg dept (dept_name, loc, budget)

Dept_name	Location	Budget
Computing Technologies	Techpark	7000000
Networking and Communication	Techpark	4000000
Computing Intelligence	University Building	6000000
Data Science and Business Systems	University Building	3000000
Mechatronics	Hitech	3500000
Electrical Engineering	Main Building	2000000

Section Table

Course_code	Sec_id	Semester	Year	Location	Room_no	Time_slot_id
18CSC303J	A1	EVEN	2022	Techpark	TP801	A
18CSE456T	A1	ODD	2021	Techpark	TP706	B
18CSE390T	B1	EVEN	2022	University Building	UB4001	C
18CSC205J	B1	EVEN	2022	University Building	UB5002	B
18CSE344T	B1	ODD	2021	Techpark	TP403	D
18CSC305J	A1	ODD	2021	University Building	UB1201	E
18CSE459T	A1	EVEN	2022	University Building	UB1210	G

Faculty Teaching

Faculty_id	Course_Code	Sec_id	Semester	Year
100186	18CSC303J	A1	EVEN	2022
100181	18CSE456T	A1	ODD	2021
100199	18CSE390T	B1	EVEN	2022
100201	18CSC205J	B1	EVEN	2022
100210	18CSE344T	B1	ODD	2021
100212	18CSC305J	A1	ODD	2021
100300	18CSE459T	A1	EVEN	2022

- Key in rel. Model

1. Super key : set of attr. that uniquely identify a row
2. Candidate key : Minimal super key
3. Primary key : Chosen candidate key
4. Foreign key : primary key of one table referenced in another table

- Relational Query language

  - Used to req into from db
  - Procedural - specifies steps to retrieve data
    eg (relational algebra)
  - Non procedural - specifies what data is needed n not how to get
    eg (SQL, Relational Calculus)