50. manu...

51. ~~In~~ zoom, ~~tab~~ view tab

Database Management Systems

Database : structured and organized collection of data i.e stored electronically as a single unit

Database system : a broader concept that encompasses (envi for mnging data) not only the data itself but also the software, hardware, users involved ~~invol~~ managing and interacting with the database.

includes all components required to store, retrieve and manipulate data effectively within an organization

Database management system:

→ s/w application that enables users to efficiently create, maintain and interact with the databases.

→ acts as intermediary between user/app and actual database ensuring data integrity, consistency and security.

Ex: MS ACCESS, DBASE IV or V

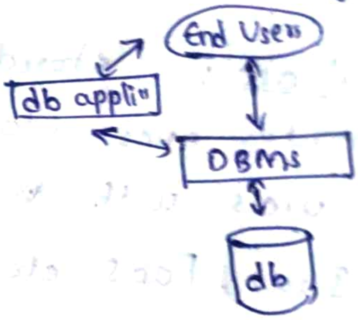
datum :- unit of data

Why use DBMS

- ✓ integrity, independence, security, abstraction
- ✓ for concurrent access to data, data recovery from Crashes
- ✓ uniform data administration
- ✓ efficient use of data
- ✓ to use user-friendly declarative query language

components of DBMS

db
dbms
end users
db applications



disadv : cost of h/w and s/w
size
complexity
higher impact of failure

Types of databases

1. Centralized db : stores data at a centralized database system. comforts the users to access stored data from diffⁿ locations through several appliⁿ.
   Ex: central library
2. Distributed db : data is distributed among different db systems of an organization.
   Ex: HBase, Apache Cassandra
   2 < Homogeneous DDB — execute on same OS under same appⁿ process and use same and carry same h/w devices
   Heterogeneous DDB — execute on diffⁿ OS under diffⁿ app procedures and carries diffⁿ h/w devices
3. R Db stored as tables
   invented by E.F. Codd in 1970.
   Ex: MySQL, Oracle, Microsoft SQL Server

④ NoSQL database : stores a wide range of datatypes

→ not structured

→ 4 types
- Key-value storage
- Graph database
- Wide column stores
- Doc-oriented database

⑤ Cloud database : stored in virtual environment and executes over cloud computing platform.

→ provides users with various cloud computing services (Saas, Iaas, Paas etc.) for accessing the db.

Ex: Microsoft Azure
AWS
Kamatera

⑥ Obj-oriented databases - uses obj-based data model approach for storing data in database system.

data is stored as objects ≈ objects in OOP language
and rep

⑦ Hierarchial database : stores data in form of parent-children relationship nodes.

it organizes data in a tree-like structure.

⑧ N/w databases follows n/w data model

⑨ personal databases

⑩ Operational database

⑪ Enterprise database

RDBMS

table | relation
rows | records | tuples
columns | fields | attributes

degree = no. of columns
cardinality = no. of rows

table | relation properties :
- each table - unique name in db
- doesn't contain duplicate rows
- tuples - no spec order
- attr - atomic

Domain = possible values each attribute can contain
specified using std data types

Integrity - Entity integrity (no duplicate rows)
Domain integrity (valid entries for given col)
referential integrity (rows cannot be deleted, which are used by other records)

User-defined integrity (enforces some specific business rules defined by users)

DBMS Architecture

- 1-tier architecture
- 2-tier architecture
- 3-tier

basic client-server architecture - to deal with a large number of PCs, web servers, db servers

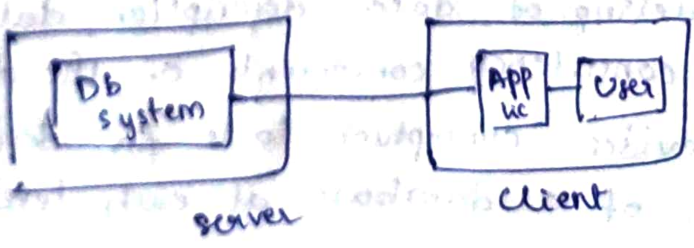
1-Tier

→ db - available to user
→ any changes done here will directly be done on db
→ no handy tool for users

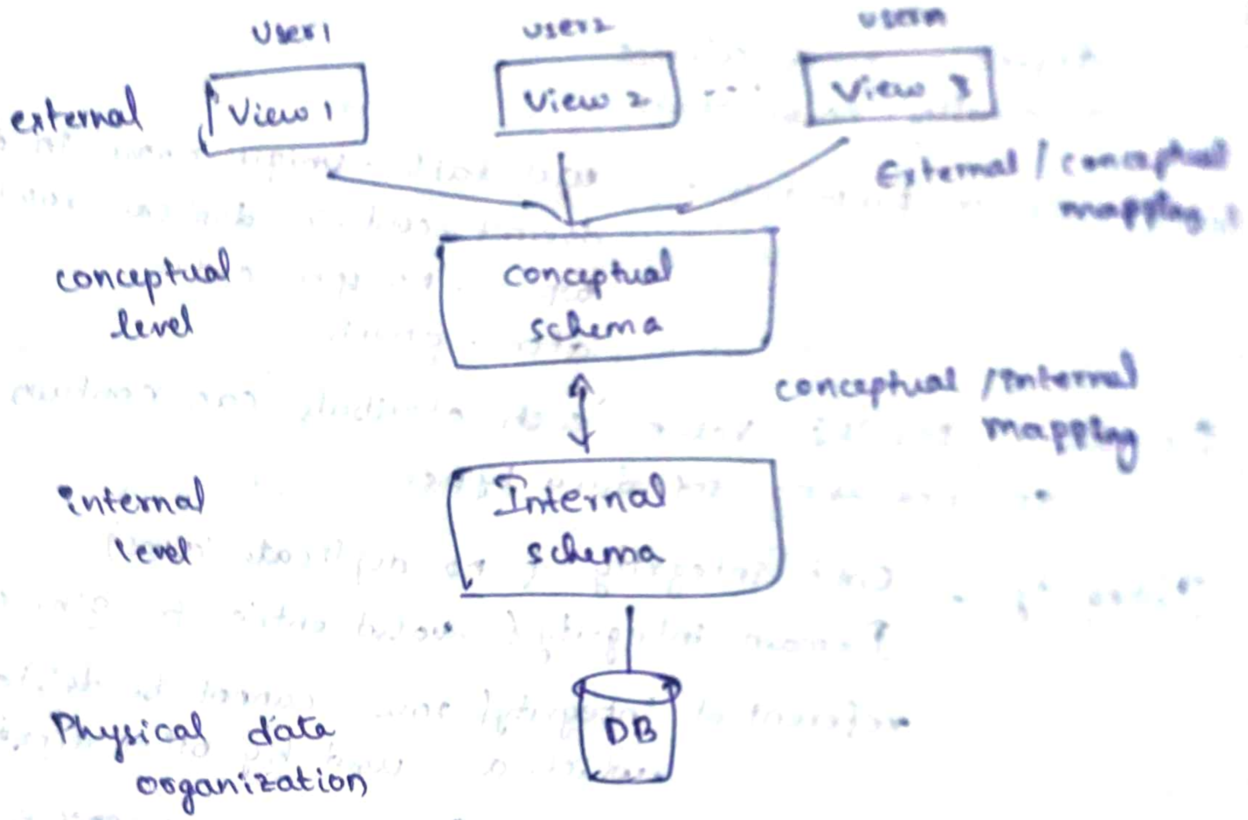
2-Tier

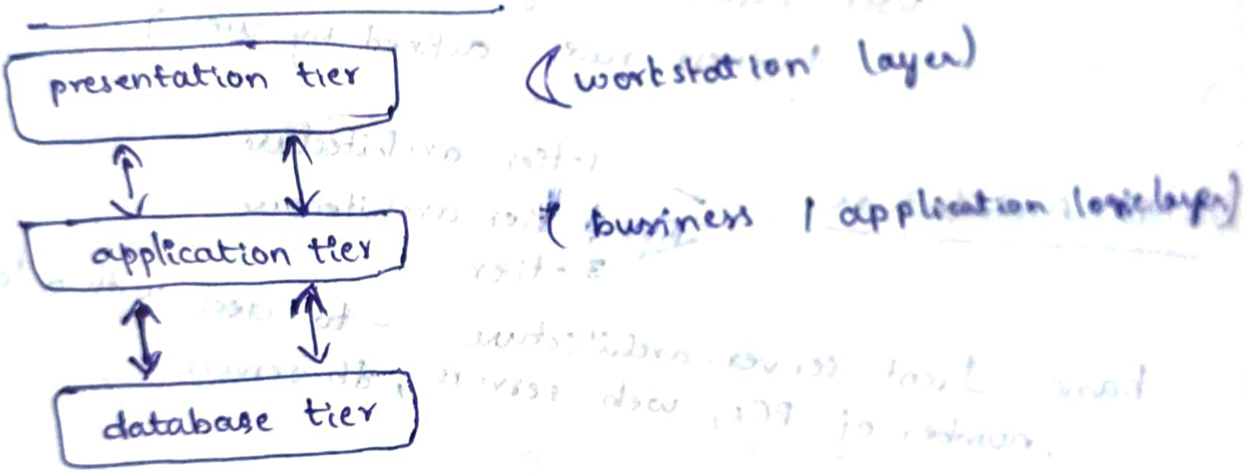
same as basic client-server archi.

appli. on client side can directly communicate with db at server side. API's are used : ODBC, JDBC.



3-Tier has another layer between client and server





Three tier architecture

Mapping - used to transform the request and response between various database levels of architecture.

Data Models

→ modeling of data description, data semantics and consistency constraints of the data.

→ provides conceptual tools for describing the design of a database at each level of data abstraction.

Types of data models
- Relational data model
- Semi structured
- E-R
- Object based

R Data model

tables - data relationships   C.F. Codd (1970s)

ER Data model - logical representation of data as objects and relationships among them.

objects ↑ entities

relationships ↑ association among entities

→ Peter Chen (1976)

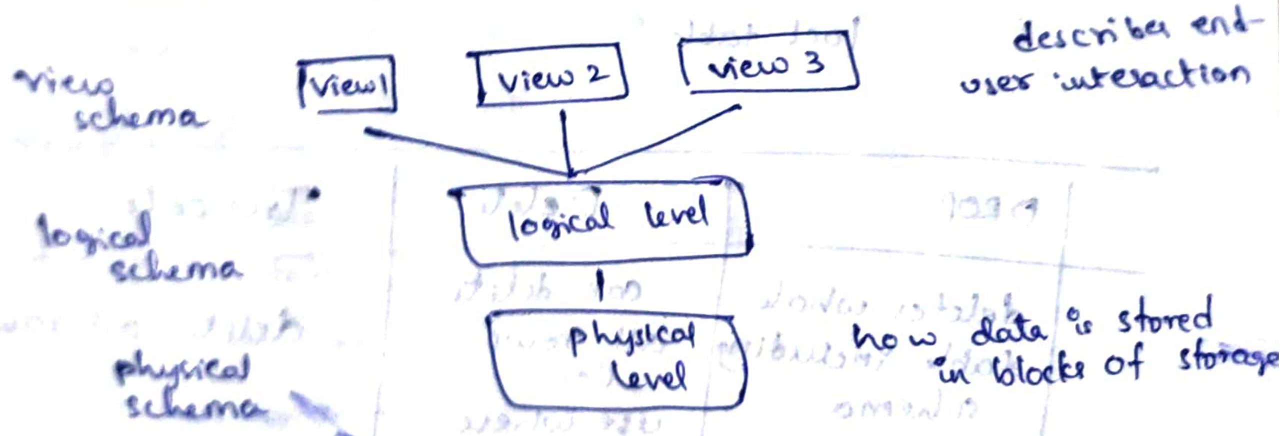
Object-based data model extension of ER with notions of functions, encapsulation, obj identity as well (1980s)

Semi-Structured data model : allows data specifications at places where the individual data items of same type may have different attribute sets.

XML used for representing semi structured data.

Schema - design of a database

overall description of the database - db schema



→ 3 levels of abstraction

logical schema

Data Independence

being able to modify the schema at one level of database system without altering the schema at next higher level.

Logical data independence
Physical data independence

Database languages

Types

DDL
- Create
- alter
- drop - delete obj from db
- truncate - remove all rec from db
- rename
- comment

DCL
- grant
- revoke

Data control lang

TCL
- commit - to save tran
- rollback - restore the db to original since last commit

DML
- select
- insert
- update
- delete
- merge (UPSERT)
- call
- lock table

	DROP	Delete	Truncate
	deletes whole table including schema	can delete all rows (or) use where	delete all rows
rollback	×	✓ can be rollbacked	×

ACID   Atomicity - defines data remains atomic

If any operation is performed on data, either it should be performed or executed completely or should not be executed at all.

Consistency   integrity of data should be maintained

Isolation -   property of db where no data should affect the other one and may occur concurrently.

Durability   - ensures the permanency of something data after the successful execution of the operation becomes permanent in the database.

U-III

ER

Relationship type - set of associations between one or more participating entity types

degree of rs : no.of participants involved in rs type

entity : object

entity type : type that is used to represent a group of objects

attributes : props of entities

Relationships 

one-to-one
one-to-many
many-to-one
many-to-many

Keys

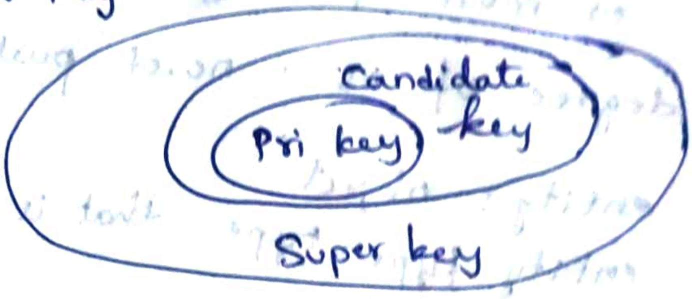
Primary key: uniquely identify the tuples in relation or table.

Candidate key: minimal set of attrs that can uniquely identify a tuple.
minimal super key
contain unique values
can contain NULL values
a table can contain ≥1 candidate keys
is " " ≥1 candidate keys

Super key - set of attributes that can uniquely identify a tuple

Adding 0/more attr to candidate key generates super key.

candidate key is super key & vice-versa ✗

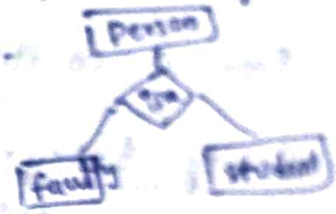


Alternate key
candidate keys other than primary keys
secondary key

Foreign key col or columns in a database that are linked to a column in different table

Generalization

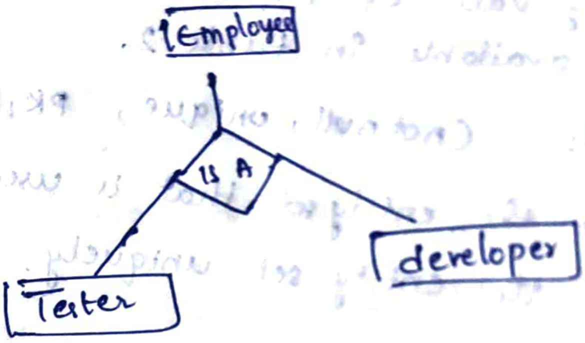
bottom up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.



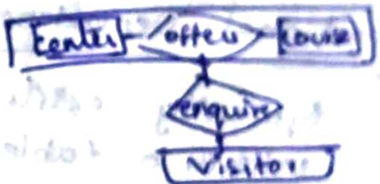
More like superclass and subclass system
but approach → different

Specialization

top-down approach
one higher level entity → 2 lower level entities



Aggregation: Relation between 2 entities is treated as single entity. Visitor enquires about both center and course



Relational algebra

- procedural query language
- step by step process to obtain result of query.

select operation $\sigma\ p(r)$

project : $\pi\ A1, A2, An(r)$

Union : $R \cup S$ — R, S must have attribute of same number

Set Intersection : $R \cap S$

Set difference : $R - S$

Rename : ρ

Integrity constraints

Domain constraint – (restrictions on the values that can be stored in a column)
Ex: age

Entity Integrity constraint
primary key can't be null

Referential integrity constraints
specified between 2 tables
if FK in table1 refers to PK of table 2,
every value of FK in table1 must be null or
be available in table 2.

Key constraints (not null, unique, PK, FK, check, default)
Keys are the entity set that is used to identify an entity within its entity set uniquely.

Functional Dependency – rs that exists between 2 attributes

typically exists between PK and non-key attribute within a table.

$X \rightarrow Y$

X: determinant
Y: dependant

Types of FD

- Trivial FD
- non-trivial FD

FD is TFD if $B \subseteq A$

if not ie. $A \cap B = \phi$, complete non-trivial.

Inference rule

1. Reflexive Rule.

   $Y \subseteq X, \quad X \to Y$

2. Augmentation Rule (partial dependency)

   if $X \to Y$, then $XZ \to YZ$

3. Transitive rule

   if $X \to Y$
   $Y \to Z$, then $X \to Z$

4. Union rule

   if $X \to Y$
   $X \to Z$, then $X \to YZ$

5. Decomposition rule

   if $X \to YZ$
   then $X \to Y$ and $X \to Z$

6. Pseudo transitive Rule

   if $X \to Y$
   $YZ \to W$, then $XZ \to W$

Normalization - process of organizing the data in db.

→ to minimize the redundancy from a relation/ set of relation

→ eliminate anomalies

→ divides larger table into smaller and links them using rs.

→ Normal form → to reduce redundancy from db table

Anomalies

insertion : when one cannot insert a new tuple into a rs due to lack of data

deletion : situation where deletion of data results in unintended loss of some other important data

Updation : when an update of single data value requires multiple rows of data to be updated.

1NF a relation is in 1NF if it contains an atomic value

2NF if it's in 1NF and all non-key attributes are fully FD on PK (eliminate partial FD)

3NF if it is 2NF
no transitive dependency

BCNF stronger 3NF
if it is in 3NF
X is super key (if $X \to Y$)

4NF if it is BCNF
no multi-valued dependency

5NF if it is 4NF
and no join dependency
joining should be lossless

Normalization

Advantages	Disadvantages
→ Normalization helps to minimize data redundancy	→ performance degrades when normalizing relations to higher NF i.e, 4NF, 5NF.
→ data consistency	→ time consuming
→ flexible db design	→ difficult if higher degree
→ enforces concept of referential integrity	→ careless decomposition X