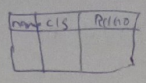
Database

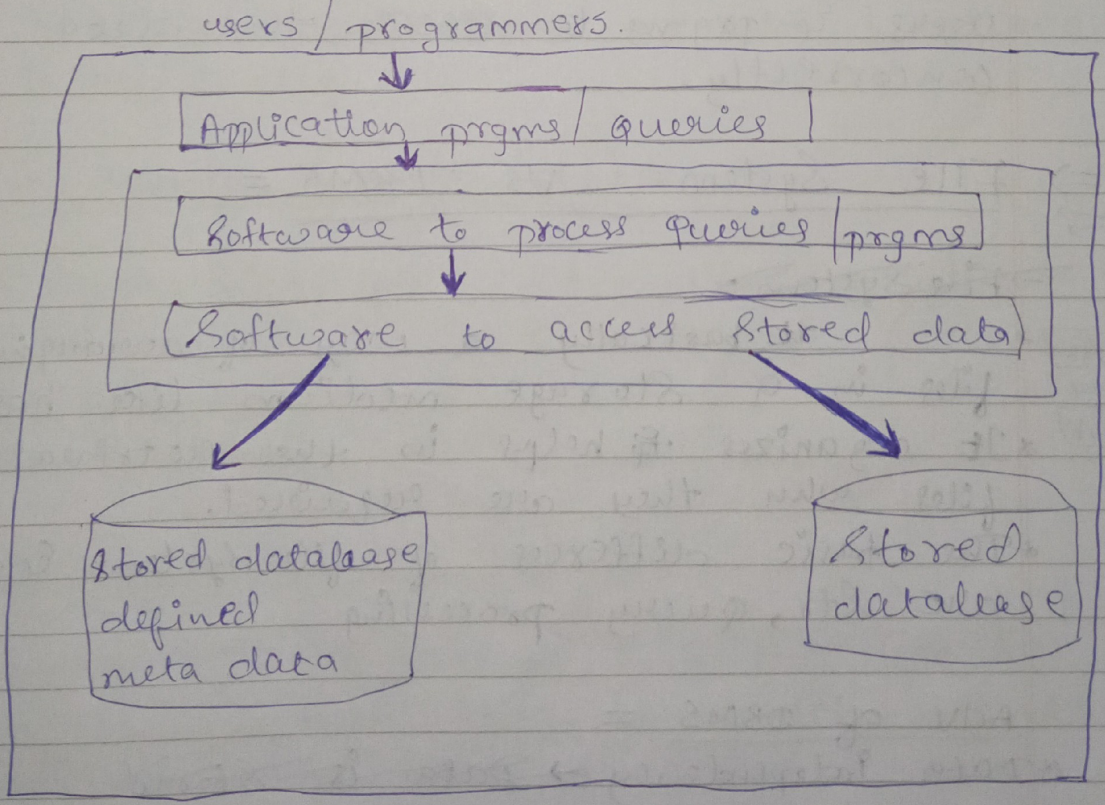


classmate
Date
Page

DBMS

* Data = Raw facts / unprocessed facts.
* Info... = processed facts / meaningful data.
* Database = It is an integrated collection of logically related data into an common pool that provides data for many application.
* Database management system = It is a collection of prgms that enables user to create & maintaine a database. Therefore a general purpose Software System that facilities the process defining, constructing, manipulating & Sharing data bhsed various applications.

*



(D) → database → easy to access data.

=> Functionality of (D) :-

1) Define → Defining a (D) involves specifying the datatypes, str. & constraints for the data to be stored in the (D).

2) Construct → (adding data)
Constructing the (D) is the process of storing the data itself on some storage medium (i.e) control by DBMS.

3) Manipulating a (D) → It includes such (s as querying the (D).
(update, alt, querying)

4) Sharing the (D) → It allows multiple users & prgms to access the data concorrectly.

=> File System VS DBMS =

~~File System~~

* It basically a way of arranging the files in a storage medium like harddisk
* It organizes & helps in the retrival of files when they are required.
* The basic differces of file system & DBMS are str, query processing

Adv of DBMS =
* Data Independency → ~~Data is stored~~
* Efficient file access
* Data Security

Root node → parent (N) → child node

Disadv of DBMS →
* over head cost of DBMS
* Impact of failures.

=> Database User =

1) Naive user
2) Application prgmer
3) Sophisticated user (query writing people)

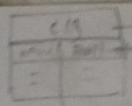
=> DBA ((d) Administrator) =
* A DBA is the info technition responsible for directing & performing all activities related to maintaining (d) enumt.
* A DBA makes sure an organisation (d)s & related applications operate functionally & efficiently.

=> (d) Models (design / str) =

① Hierarchical (m) tree like str (one to many)
② Ntwrk (N) graph like str (many to many)
③ Entity relationship
④ Relational

1) Hierarchical (m) =

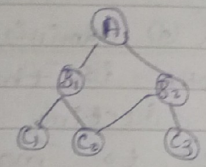
* Datas are organised in a tree like str with a single root node to which all the othr datas are linked.
* Hierarchy starts from root node & expands like a tree adding child nodes to parent nodes.



* Child node will only have a single parent node.
* It is a 1 to many relationship.

2) Network (m) =

* It is a many to many relationship.
* Datas are organised in a graph like str & are allowed to have more than 1 parent node.



3) Entity-Relationship (m) =

* Relationships are created by dividing obj of interest into entities & its characteristics into attributes.
* Diff entities are related using relationships.

Student		
Name	Cls	Rollno.

Entity
attributes.

4) Relation (m) =

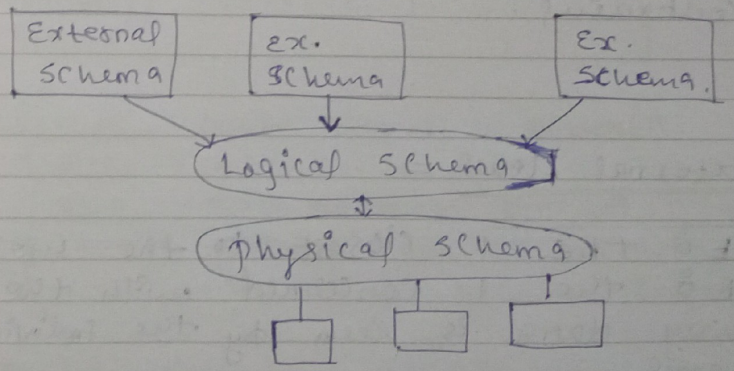
* Data is organised in 2D tables & the relationship is maintains by storing a common field.
* Basic str of data in a relational model is tables.

=> Schemas & Instance =

* Schemas = Discription of a database [overall design of a db] which is specified during (d) design & it is not expected to change frequently.
  It specifies the names of the entity & attributes & the relationship among this entities.

* Instance =
  The data in the (d) at a particular movmnt in time → (d).
  It is also → the crnt set of occurance/ instance in (d).

=> 3 Scheme Architechture =



→ Physical schema =

* Internal level also → physical level. It is the 1 closest to physical storage.
* It is the 1 concern with the way the data is physically stored.
* The internal level has an internal schema which describes the physical storage str of db.
* The internal schema uses a physical data model & describes the complete details of data storage & access paths for the db.

→ Logical schema / Conceptual schema =

* The conceptual level has a conceptual schema which describes str of the whole database for a community of users.
* The conceptual schema hides the details of the physical storage str & concentrates on describing entities, relationship, user operation & contraints.

→ External level =

* It is the 1 closest to the users
* It is the 1 concerned with the way data is seen by the individual users

⇒ Data Independence =

* capacity to change the schema at 1 level without having the change the schema at the higher level → D.I
* 2 types of D.I →
  a) Logical D.I = used to separate external from the logical level
  b) physical D.I = used to separate logical levels from physical level.

⇒ Database Language =

Data def lang	Data manipulation Lang	Data control lang	Transmission control lang.
1) CREATE	SELECT	GRANT	COMMIT
2) ALTER	INSERT	REVOKE	ROLL BACK
3) DROP	UPDATE		
4) TRUNCATE	DELETE		
5) RENAME			

str
- create → create obj in db.
- Alter → Alter the str of db
- DRop → delte obj from db
- Truncate → remove all rec from table.
- Rename → rename an obj

I Data def lang = (DDL)

(t) → table

* DDL → Data Def Lang.
* used to define (d) str / pattern
* used to create schema, tables, Indexes, etc.

1) CREATE → used to create objects in (d).

II Data manipulation Lang = (DML)

* DML → Data Manip Lang.
* used for accessing & manipulating data in (d).

1) SELECT → retrive data from (d).
2) INSERT → Insert data into a ~~(d)~~ table.
3) Update → update existing data within a table.
4) DELETE → dlt a record frm (t).

III Data Control Lang = (DCL)

• (giving permission in (d).)
* used to control privilage in a (d).
1) GRANT → gives user access privilage to the (d).
2) REVOKE → takes back permission from user.

[grant → permission കൊടുക്കാൻ
revoke → " deney cheyyan.]

IV Transmission Control Lang = (TCL)

* used to run the changes made by DML statemnts.

1) COMMIT → Saves the work done (change ചെയ്തതിൻ ശേഷം കൊടുക്കാൻ)
2) ROLLBACK → Restores the (d) to the original (change ചെയ്യുന്നതിൻ മുൻ ഉണ്ടായിരുന്നതിലേക്ക് തിരികെ പോകും)

=> Interface = (I) (5 types)

* medium used to interact with DBMS.

1) Menu-based (I) = (menu select ചെയ്യാൻ)

* This (I) prsnt the user with list of options → menus

2) Form-based (I) = (form type)

* It displace a form to each other.
* users can fillout all forms entries to insert a new data, they can fillout certain entries

3) Graphical user (I) = (GUI)
(diagramatical (I))

It typically displace a schema (str of database) to the user in diagramatical form.

4) Natural lang (I) =

It accepts requests written in eng / some other lang.

5) Speech input & output =

limited use of Speech as an input query & speech as an answer to the question / result of request is becoming

(E) → Entity

common place.

⇒ Basic concept of Entity relationship Model :-

→ **Entity** :-

* It is a thing / sub in the real world that is distinguishable from all other obj.
* It has set of properties & the values for some set of properties may uniquely identify an entity.

→ **Attributes** :- (A).

* An (E) is represented by a set properties → (A).
* (A) are discriptive (pro) processed by each member of an (E) set.
* 6 types →
  1) composite (A) (name → 1st letter, 2nd letter)
  2) Simple (A) (A) cannot subdivide (A).
  3) Single valued (A)
  4) Derived (A)
  5) multi valued (A)
  6) Stored (A)

(1) Composite (A)	(2) Simple (A)
(A) that can be divided into further (A)s. eg → Name can be divided by 1st name, 2nd name.	(A) that cannot be divided into further (A)s. eg → Salary cannot be divided to other.

(3) Single valued (A)	(4) Multivalued (A).
Most (A)s have a single value for an particular (E). Such (A)s → S.V.(A). eg → age is a single valued (A) of a person.	It can have a set of values for an particular (E). eg → language of a person.
(4) Derived (A)	(5) Stored (A).
(A) that can be derived from other (A)s. eg → age of a person.	(A) from which the value of other (A)s are derived. eg → DOB of a person.

⇒ Entities =

* **Types** →
  1) Tangible → Physically
  2) Intangible → Logical

* **Entity Set** = Grp of entity having some (A).

* Entity type = Representing entity with a name.

eg → Student

Rollno	name	Branch
101	Sam	CS
102	Ram	bCA
103	Sam	bbA

→ fig 1.1

① entity set → Student.

② entity type →

101	Sam	CS
103	Sam	bbA

(having common entity)

* Entity type → 2 types.

a) Strong entity type = (E) type having key attribute. (unique no.)

b) weak E. type = (E) type not having a key (A).

eg → ① Strong E-type → 101, 103 (fig 1.1)

② (weak E-type)

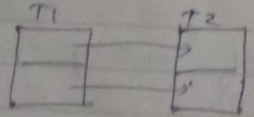
name	branch
Sam	CS
Ram	BCA
Sam	BBA

→ eg for weak E-type. (no key (A))

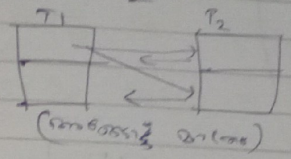
fig 1.2

⇒ Relationship in table =

① one to one.

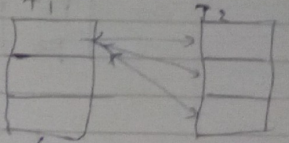


② one to many



(one to many)

③ ~~one~~ many to many

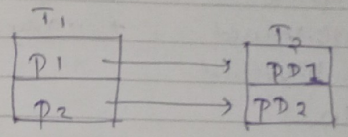


(many to many)

I one to one =

* If 1 record of T_1, is only related to 1 record of T_2.

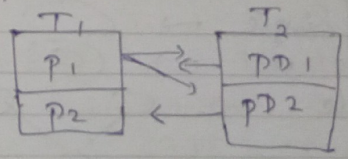
* eg →



PD → passport detail

II one to many =

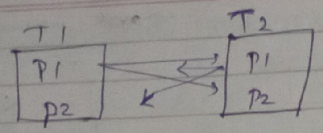
* 1 rec of T_1 is related to multiple records of T_2.
* The rec of T_2 is related to only 1 rec of T_1
* eg →



III many to many =

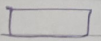
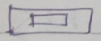
* 1 rec of T_1 is related to multiple rec T_2.
* 1 rec of T_2 is related to multiple rec T_2.

eg →

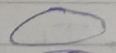
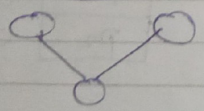
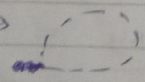


Components in Entity relation

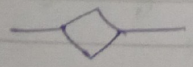
Entity	Attribute	Relationship
* Strong entity	* Simple	* 1 to 1
* weak entity	* composite	* 1 to many
	* Single-valued	* many to many
	* multi-valued	
	* Stored	
	* Derived	

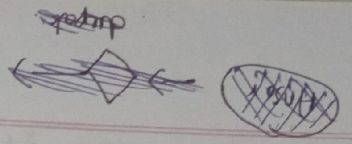
→ Notations → Strong Entity → 
weak Entity → 

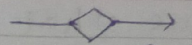
→ Attributes notations –

* Simple / Single → 
* multi valued → 
* composite → 
* Derived → 

→ Relationship notations →

* 1 to 1 → 



* 1 to many → 
* many to many → 

→ Keys in DBMS =

1) candidate
2) primary
3) Alternate
4) Super
5) foreign

eg →

E_id	E_name	E_adhar	email
101	John	2103	--
102	AK	21045	--
103	BK	21056	--

null ಇರಬಾರದು not null

① * candidate → E_id , E_adhar email (unique key)
② * P.key → (C.key - ನಲ್ಲಿ ಇರುವಂತಹ ಒಂದು key)
↳ (only 1 in a table) → E_id , E_adhar [not null]
(email ಇರಲೇಬೇಕು ಅನ್ನುವಂತಿಲ್ಲ)

1) candidate key →

* It is a subset of a Super key set where the key which contains no redundant attribute is none other than a candidate key.
* Role of C.key is to identify a table row / column uniquely.

2) primary key →

* It is a column / columns containg

value that uniquely identifies each row in a table.

3) Alternate key =

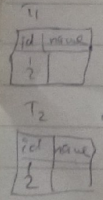
eg → [illegible] → email
(c-key → [illegible] p·key → [illegible])

* Exacpt p·key all c·keys will be considered as A·keys.

4) Super key =

* combination of columns that uniquely identifies any row with in a relational database managment system.
* A c·key is closely related. Concept where the super key is reduced to the min no of column required to uniquely identify each row.

5) Foreign key =



* It is a field in one table that refers to the p·key in another table.
* The table with the f·key called as the child table & the table with the p·key → parent table.

⇒ Constrains in DBMS =

* Constrains are the set of rules that ensures whether an authorised user modifies the database they do not disturb the data consistancy.
* constrains enforce limit to the data or types of data that can be inserted, updated, deleted from the table.

→ Types of constraints = (con)

a) key (con) (unique [illegible])
b) Entity integrity (con) (null [illegible])
c) Domain (con)
d) Referencial integrety (con).

a) key (con) =

* All values of the p·key must be unique (i.e) that no 2 tuples can hve the same combination for all their attributes

b) Entity integrity (con) =

* States that the p·key value cannot be null.
* we use the p·key 2 identify individual rows in the table

c) Domain (con) = (values in the table → domain)

* It defines the domain / set of values for an attribute.
* It specifies that the value taken by the attribute must be atomic value from its domain.

id	name	cls.

→ Table / relation
id, name, cls — 3 attri / field
each rows — tuple / rows
cols / domain.

d) Referencial integrety (con) = (like f key)

* This (con) is enforced when a f. key references the p. key of a relation.
* It's specifies that all the values taken by the f. key must be available in the relation of the p. key / be null.

⇒ Relational Model =

Retrieve the data.

Procedural lang.	Theoritical	practical
	* relational algebra	* SQL.
	* " calculus	

Relational Algebra.

Unary option	set theory option	Binary option	extended (derived op.)
* selection (σ)	* union (∪)	* Join (⋈)	* outer join
* projection (π)	* Intersection (∩)	- Inner join	- left outer
* Rename (ρ)	* Cross product (×)	- Theta join	- right outer
	* minus / set difference (−)	- Naturally join	- full outer
		- division	* Aggregate (G)

selection → not table - row values select only.
↓
horizontal subset. → σ

I Unary operation =

1) Selection =

* Sigma (σ)

* Horizontal subset. [we get rows full]

Selection syntx → syntx → σ (condition) (relation name)

eg →

Student.

id	name	%
1	AK	90
2	bk	80

σ (% > 80) (Student)

o/p. → | 1 | AK | 90 |

2) projection (π) =

* pi (π)
* vertical subset.

Syntx. → π col1, col2 ... (relation name)

eg → π id, name (Student)

↳ I get id & name in vertical data

→

1	AK
2	bk

eg → π id, name σ (% > 80) (Student)
(condition)

→ | 1 | AK |

3) Rename (ρ) =

* Rnto (ρ) , change table name.

Syntx → ρ (newname) (relation name)

eg → ρ (final) (Student)

Student → new name
↳ final

1) Set theory operation =

~~Union (∪)~~ eg →

R

id	name
101	C++
102	DBMS
103	Py

S

id	name
101	C++
104	C
105	Java

1) Union (∪) =
→ R∪S

eg →

[combination & no duplicates]

R∪S

id	name
101	C++
102	DBMS
103	Py
104	C
105	Java.

* Union operation b/w 2 relation R & S denoted by R∪S returns a relation instance containing all tuples that occurs either in relation instance R / S / both.
* R & S must be union combinable i.e the schema of the result is defined to identical to the schema of R.

2) Intersection (∩) =

eg → R∩S.

id	name.
101	C++

* Intersection option b/w 2 relations R & S denoted by R∩S returns a relation instance containing all tuples that occurs both in R & S relation.

3) Set difference | minus = R-

R - ഉള്ളതും , S - ഇല്ലാത്തത്.

eg → R-S

id	name.
102	DBMS
103	python

S-R

id	name
104	Java
105	C

* S. diffrnce opration b/w 2 relation R & S denoted by R-S returns a relation instance containing all tuples that occurs in R ~~& S relation~~ i.e not in S.

4) Cross product (×) = (multiply)

eg → R×S.

①

id	name
1	A
2	b

id	name.
3	K
4	C.

2×2=4 rows

id	name	id	name.
1	A	3	K.
1	A	4	c
2	b	3	K
2	b	4	C.

②

R

id
1
2
3

③

S

id
4
5.

②

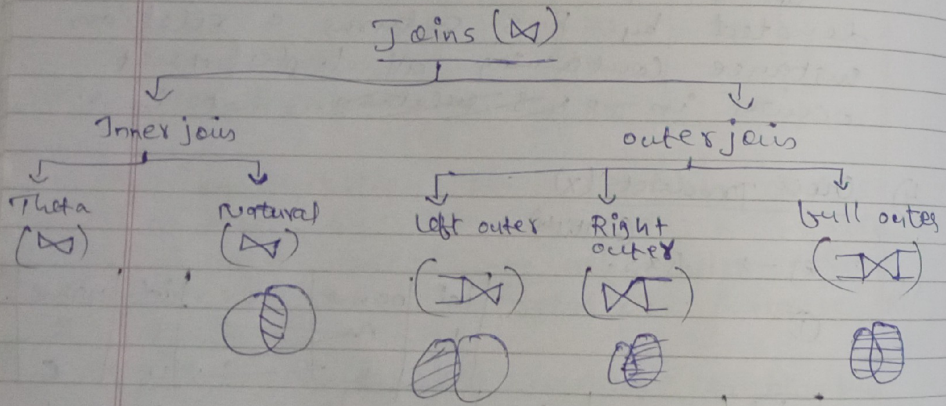
3×2 = 6.

R×S →

id	id
1	4
1	5
2	4
2	5
3	4
3	5

* cross product / cartesian product operation b/w 2 relation R & S denoted by R x S returns a relation instance whose schema contains all attributes of the relation instance S followed by all attributes of the relation instance S.
* Thus output relation R x S as a schema which is the concatenation of R & S.

=) Joins in DBMS =



1) Theta join =

eg →

C

car model	price
C1	4000000
C2	5000000
C3	8000000

J

Jeep model	Price
J1	6000000
J2	10000000

J ⋈ J price > C price (condi-)

C ⋈ J

Jeep m	price	carmodel	price.
J1	6000000	C1	4000000
J1	6000000	C2	5000000
J2	10000000	C1	4000000
J2	10000000	C2	5000000
J2	10000000	C3	8000000

$C_3 \times J$ — $J > C$ false.
$C_3 \checkmark$ b C2 ← J_2 — $J > C$ True.

2) Natural Join =
relation will get join with common attribute

eg →

S

id	name	Dept
101	AK	1
102	bK	2
103	CK	3
104	DK	1

D

Dept	Dname.
1	CSE
2	ECE

id	name	Dept	Dname.
101	AK	1	CSE
102	bK	2	ECE.
104	DK	1	CSE

3) Left outer join =

resultant relation will have all tuples of S, & common tuples of D.

S

id	name	Dept.
101	AK	1
102	bK	2
103	CK	3

D

Dept	Dname.
1	CSE
2	ECE.

id	name	Dept	D name
101	AK	1	CSE
102	bK	2	ECE
103	cK.	3	NULL

(3 ఇక్కడ So null)

2) <u>Right outer join</u> =

S D