DBMS

A DBMS is the collecⁿ of interrelated data & a set of programs to access those data. The collecⁿ of data, usually referred to as the database, contains inforⁿ relevant to the enterprise. It is a software package ~~used~~ to design to define, manipulate, retrieve & manage the data in a database. A DBMS generally manipulates data itself, the data fields, file structures. It also defines rules to validate & manip-ulate this data.

Functⁿ - (1) Data Dictionary management:

One of the most imp. functⁿ of DBMS is the DDM. DBMS stores definitⁿ & their relationships (meta data) in a data dictionary. So, all programs that have access to the data in the database work through the DBMS. It uses DD to look up for the req. data compo. with relieves us from coding the complex relⁿ in each pregram. Any changes made in the DBS. are automatically recorded in the DD, thereby provides DI, remove Dep. from system.

(2) Data storage management: The DBMS creates & manages the complex struct. req. for data storage, thus relieving us from the difficult task of defining & prog. the phy. data chara. of

DSM is imp. for DBMS Perfo. Tuning. Perfo. Tuning relates to the activities that make the D.B. perform more efficiently in terms of storage & speed.

(3) Data Transform" & Rep: The DBMS transforms the entered data into the req. data structure. The DBMS relieves us from the chore of making distinction b/w the phy. data or log. D. Format. i.e. DBMS formats the physically retrieved data to make it confirm to the user's expectat". for eg - Imagine an enterprise D.B. used by a multinational company. An end user in England would expect to enter date such as 11/07/2009, in the contrast same date would be entered in U.S. as 07/11/2009. The DBMS syst must manage the dat in proper format for each country.

(4) Security Management: This is another imp. funcn of DBMS. The DBMS creates the security system that enforces users security and data privacy. Security rules determines wch users can accen the data, wch data can be accened & wch opern can be performed (delete, add, modify or read). This is spclly imp. in the multiuser D.B. syst.

(5) Multiuser Access Control: To provide data integrity & data consistency, the DBMS uses sophisticated algo's that ensures multiple users can accen the D.B. concurrently without compromising the integrity of the D.B.

(6) Back & Recovery: DBMS provides back-up & Recovery to ensure data safety & integrity. Recovery management deals with the recovery of database after a failure.

(7) Data integrity management: DBMS promotes & enforces the data integrity rules, thus minimizing data redundancy & maximizing data consistency. Data Rel" stored in D.D. are used to enforce D.I. Ensuring D.I is imp. in Transac"-oriented D.B & Sy

(8) D.B. Access lang: DBMS provides data access through a query lang. A Query lang. is a non procedural lang. one that lets the user specify what must be done without having to specify how it is to be done.

Advantages: (1) Improved data sharing: DBMS helps create an env" in wch the end users have accen to more & better managed data. Such accen makes it possible to the end users

to respose quickly to the changes in their environment.

(2) Improved data security: The more users access the data, the more is the risk of data security breaches. Corporat" invest considerable amt. of time, money to ensure corporate data are used properly. DBMS provides a framework for the better enforcement of data privacy & security policies.

(3) Minimized data inconsistency: Data inconsistency occurs when diff. versions of same data appear at different places. eg., data inconsistency occurs when a company's sales dpt. stores sales rep. name as "Bill Brown" & Personnel dpt. as "Mr. Brown".

(4) Disadvantages -

(1) H/C cost - DB requires sophisticated hardware & software and highly skilled personnel. The cost of maintaining the hardware, software and the personnel required to operate & manage the DB can be substantial. Training, licensing, regulat" compliance costs are often overlooked when DBS are implemented.

(2) Complexity - The provision of the functionality that is expected of a good DBMS makes the DBMS an extremely complex piece of software. Database designers, developers, DBA's & end-users must understand this functionality to take full advantage of it. Failure to understand this functionality may lead to bad decision which can have serious consequences for an orgn.

(3) Size - The complexity & the breadth of functionality makes DBMS an extremely large piece of software occupying many megabits of disk space & acquiring substantial amt. of memory to run efficiently.

(4) Performance - Typically, a file based system is written for a specific app. such as invoicing. Hence the perfo. is good. However, DBMS is written to cater many app's which in turn may lead to many app's not working as fast as they used to.

(5) Higher impact of failure - The centralizat" of resources i/c the vulnerability of the system. Since all users & app's rely on the viability of DBMS, the failure of any component can bring operat" to a halt.

(6) Cost of Conversion -

User Interfaces:

(1) Naive users - are those who need not to be aware about the presence of database in the system or any system supporting their usage. These are the end users of DB who work through a menu driven app. prog.

(2) App - prog - are the professional programmers who are responsible for developing programs or user interface. The app. prog. can be written in G.P. prog. lang. or the commands available to manipulate a D.B.

(3) Sophisticated users - interact with the system without writing the programs, infact they form their request in a DB Query Lang. They submit each such query to the Query Processor whose process is to break down the DML inst to statements to inst. wch is understood by the storage manager.

(4) Specialized users - are the sophisticated users who write specialized data B. app. that donot fit into the traditional D-P framework.

Database Administrator:

One of the main reasons for using DBMS is to have control of both the data & programs that acts to those data. A person who has such central control is called DBA.

Functions:

1) Schema Definitn - The DBA creates the original database schema by executing the set of DDL st. in DDL.

(2) Storage struct & access method defining.

(3) Schema & the phy. org. modifn - The DBA carries out the changes in schema & org. in order to reflect the changes in orgn.

(4) Granting authorizn of D.A - DBA can regulate wch part of DB the various users can acss.

(5) Routine maintenance -

- periodically backing up the D.B.
- updating disc space as req.
- monitoring jobs running on D.B & ensuring perfo. isn't degraded.

Difference b/w file & DBMS:

(1) DBMS co-ordinates both the phy. & the logical access to the data where as file sys. co-ordinates only the phy. access.

(2) DBMS is designed to allow flexible

access to the data (i.e. Queries) whereas file sys is designed to allow predetermined access (i.e. compiled prog.).

3) Redundancy is controlled in DBMS but not in the file system.

4) Unauthorized access is restricted in DBMS but not in file sys.

5) DBMS provides back & recovery but not in file sys.

6) DBMS provides multiple user interfaces. Data is isolated in file system.

7) DBMS reduces the amount of duplication by ensuring that a phy. data is available to all prog. authorized to have access to it. while in file system the data written by one program is not readable by another.

8) "File manager" is used to store all rels in dirs in FS. whereas DBA stores the ~~file~~ rels as the stru. table

9) File sys. Transac aren't possible in DBMS, are possible.

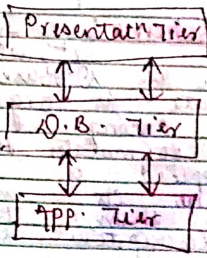
10) Saves in tempo. locan whereas in D.B. in well arranged & permanent locan.

Architecture of DBMS: The design of DBMS depends on its archi. It can be centralized or decentralized or hierarchical. The arch. can be seen as either single tier or multi tier. An n-tier archi. divides the whole sys into related but independent n modules.

✓ 1-tier: in this, DBMS is the only entity, the user directly sits on the DBMS & uses it. Any changes done here will be directly done on the DBMS. It doesn't provides any handy tools to the end users. D.B Designers & the programmers normally prefer to use single tier archi.

✓ if archi of DBMS is 2-tier then it must have an app through wch DBMS can be accessed. Prog. use 2-tier where they access the DBMS by means of app. Here the app. tier is entirely independent of D.B. in terms of oper, design & prog.

✓ 3-tier Archi: It separates its tier from each other in terms of the complexity of the users & how they use & the data present in D.B. It is most widely used archi. to design DBMS.



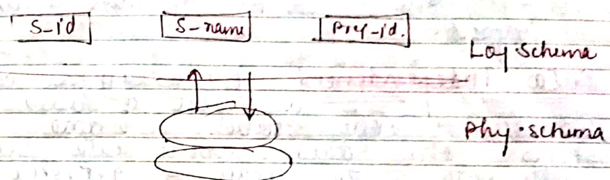
(a) **Database Tier:** At this level the database resides along with its Query Processor lang.. We also have the reln that defines the data & their constraints at this level.

(b) **App. Tier:** At this tier resides the app. server & the programs that acess the D.Base. For the user this tier presents the abstract view of D.Base. End users are unaware of any existence of the Database beyond app. At the other end, the D.B. tier is not aware of any other user beyond the app. tier. Hence, app. tier sits in the middle & acts as mediator b/w the end user & the database.

(c) **User (presentatn) Tier:** End users operate on this tier & they know nothing about the existence of the D.B. beyond this layer. At this level multiple views can be provided by apps of D.B.

Database Schema: It is the skeletal structure that rep. logical view of D.Base. It defines how the data is organized & how the reln among them are formulated. It formulates the constraints to be applied on the data.

It define its entities & the reln among them. It contains the description details of Data Base which can be depited by using the schema diagrams. Its D.B. designer who design the schema to help prog. understand the D.B & make it useful.



- **Phy. Database schema:** ~~It defines~~ This schema pertains to the actual storage of the data & its form of storage like files, indices etc. It defines how the data will be stored in a sec. storage.
- **Logical D.B. Schema:** It defines all the logical constraints that need to be applied on the stored data. It defines tables, views & integrity const.

Database instance: Database schema is the skelton of D.Base. It is designed when the D.Base doesn't exist at all. & D.B.S. doesn't contain any d/Info. ... is diff. ... it is ... to make changes.

Database instance is a state of operational D.B. with data at

any given time. It contains snapshot of the D.Base, it tend to change with time. & DBMS ensures that every instances is in a valid state by following all the validatn constraints that the D.B. designers have imposed.

Data Independence: A database contains a lot of data in additn to the user's data. It stores the data about data called the meta data, to locate & retrieve the data easily. It is rather difficult to update or modify the meta data once it is stored in database. But as the DBMS expands, it needs to change over time in order to satisfy the user's requirement. If the entire data is dependent it'll become tedious & highly complex job.

Metadata itself follows a layered archi. so, that when we change the data at one level, it doesn't affect the data at another level. This data is independent but mapped to each other.

Logi. Data indp.

Logi. schema

Phy. schema

Phy. D indp.

Log. indp - contains infon about how is data managed. eg. Relⁿ stored in DB & all constraints app. to it.

It is a kind of mechanism, wch liberalizes itself from the actual data stored on the disk. If some one do some changes on the disk, then it should not change the data residing on the disk.

Physical Data Independence: All the schemas are logical, & the actual data is stored in terms of bits in the disk. It is the power to change the physical data without impacting the schema.

eg: In case we want to change or upgrade the storage sys. itself - it should not have any impact on the schemas.

Attributes: Each entity of the E-set is distinguished by the prop. or chara. These distⁿ prop. of the E-set are called as the attributes. So, the attri. are the collectn of prop. of entities. A set of enⁿ is categorized by the set of attri. All the attributes has values. eg: a student entity has name, age, class as attri.

- **Base Value** - Entering the values for these attributes is compulsory. eg: D.O.B., R.No., marks.

Derived Attri:- These attri. don't have any existence in the phy. D.Base & their values is derived from the base attri. eg. Total marks, salary, D.O.B.

Atomic Attri:- The attri. that can't be further divisible into some more attri., atomic attri.
eg. the basic salary, sex, marks in parti. sub, rank.

Composite Attri:- The attri. that can be further divisible into some more attri., composite attri. or in other words, the combin^{n} of more than one attri. is called as the composite attri.
eg. a student's name can be further divided into first name, middle name, last name, similarly, address can be divided as the PIN no., state no., city, state.

Single value Attri:- The attri. that takes only one single value, single value attri.

Multivalue Attri:- That takes more than one value.
eg. Colour of the flag.

Keys:- A key is the attribute or the attri. of the entity which uniquely identifies the entity. It means that value of that entity-set is not repeated. It never comes more than once in the complete E-Set.
eg. S_id, Ofle no.
If there is more than one key attri. in the E-set, then the keys are termed as the candidate key. All the attributes that have the feature of being unique a key attri.
A primary key is one of the candidate key chosen by the database designer to uniquely identify the entity set. A primary key is not permitted to store a null value. Without a primary key, an entity never exists in E-set.
Sometimes in an entity set there is no attri. wch can become key attri, then we combine the several attri. & make a group attri. as keys. That group is called as the compo. key. The attri. contributing in the comp. key can't have NULL value.

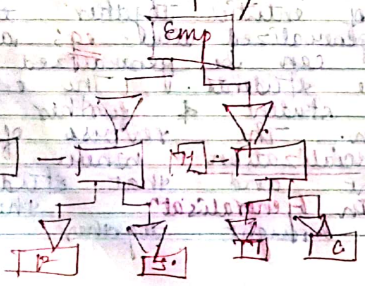
Generalisation & Specializatn: Generalisation is the process of clubbing of entities together to represent a more generalized view. eg; a particular student Mira can be generalized along with all the students. The entity shall be student & further student is a person. The reverse of this is a the specialization where a person is a student and that student is Mira. In generalisatn the no. of entities are brought together into one

based on their similar characteristics. It always uses a relationship of "is a"

eg: Employee is the generalisan of classes of objects cook, waiter., cashier etc. It always follows "is a" relnship therefore manager is a emp, cashier is emp. they all are clubbed into a single class emp.

Generalisatn process can be pictorised as - here the entity set employee is the generalisation of full time employee & part time employee.

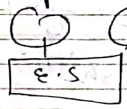
Specilizatn is the reverse process of Generalisan. It is the abstracting process of introducing new char to the existing class to create one/more new class of obj. A group of entities in specializn can be categorized into sub-grps. based on their char. It follows top-down app. in wch one higher level entity can be broken down into lower entity. It defines one/more subtypes of supertypes thus forming sub/super type rel.



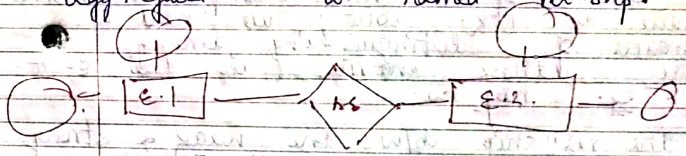
• Aggrigat"n is also an additional feature of E-R model. It is an abstrain mechanism. It is the collaboran of elementary items, attrib. or grouping of E-set, relnship sets etc.

3 typis -

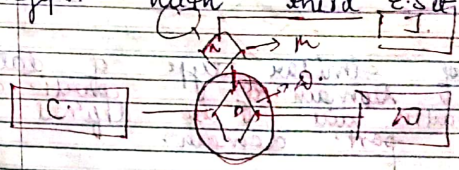
(1) When the attri. are aggregated & i'd. in to an entity set.



(2) When the E.S. and the associatn b/w them is identified and then they are aggregated a named relnship



(3) The associan or the relnship can also be agg. into another relnship by agg with third E.set.

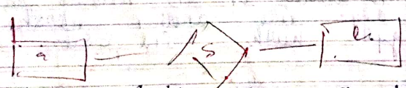


Weak | Strong E-sets The entity set which does not have sufficient attri. to make a pri. key is called as the weak en. set.
An E. set that has a P.K. is called st. E. set.

eg. Let us consider an E.set. Payment, it has the attrib. Pay_no, Pay_dat, Pay_amt, although each payment is distinct but payment for diff. loans may share the same payment no.

Mem. of strong E. set is called as the dominant entity & mem of w. E. set, subordi. ent. A weak entity set does not have a P.K. but we need a means of distinguishing among all those entries in the E-set.
eg. Pay_no.

The relnship b/w the weak & strong E-set is called as the Id. reln.

#


The set of similar type of data is called Domain of partic. attri. Every attri will be defined under a part. domain.

tupple rep. an instance of an entity or reln. Every attri. takes its value from the domain spec. to it. So, the tuple is the comb. of attri. values.

In Relt Data Base Model the data is organized into tables. These tables are called Relations. A row in a table rep. a relnship among a set of values. It uses flat struct. to rep. relnship. The data model is the collec. of table.

Database.
sup (___ ___ ___ ___)
pt (___ ___ ___),
shpt (___ ___ ___).

Network model uses graph struct. with ptrs to rep. relnship.

Database Design & Normalizan.

Functional Dependency – Determinat^n of values of non-key attri. from key attri. values of a reln is called functional dependency.

- It means that the non-key attri. are determined by the key attri. or we can say that the non-key attri. are dependent on the key attri., This is called as the functional dependency.
- This is the reln^{ship} b/w the two attributes typically b/w the P.K. & the other non-key attr. with in the table.

Suppose a reln is defined as –

Reln (A1, ——— ,A2).

if only one key attri. is identified then it is primary key then the attri. is called prime attri & all the other attri. are called non-prime attri. & this dependency of prime attri. towards non-prime attri is shown –

- A1 1 ⟶ A2, A3, ——— An.
- A1 ⟶ A2
  A1 ⟶ A3.

columns can't have unique value such that multiple row. can't be used to fetch the same

Eg

Course	Content
Programming	Java, C++
Web	HTML, PHP, ASP.

We re-arrange the rel^n as below, to convert it into First Normal Form -

Course	Content
Programming	Java,
Programming	C++
Web	HTML
Web	PHP
Web	ASP.

Each attribute must contain only a single value from its pre-defined domain.

II Normal Form: A database is in the second normal form if it satisfies the following condith -

- if it is in I normal form.
- all non-key attri^s are fully functional dependent on the P.K.

In a table, if attri^ B is functionally dependent of A, but is not functionally dependent on the subset of A then B is considered as fully dependent on A.

Hence, in 2NF all the non-key attri^s can't be dependent on the subset of the primary key.

S_id	Purchase Loch
1	UP
3	HP
1	TP
2	Assam
3	
4	

This table has composite primary key [C_id, S_id] & a non-key attri^ [Purchase Loch].

In this table the Purchase loch depends only upon the S_id, wch is only the part of the primary key. Therefore this table doesn't satisfies the 2nd normal form.

To bring it in the 2nd normal form. we break the table into 2, & now we have 2 tables -

C_id	S_id
1	1
1	3
2	1
3	2
4	3

Table Purchas

S_id	Purchase Loch
1	
3	
1	
2	
3	

Table store.

We done this to remove the partial functional dependency that we had.

Initially. Now, in the table [Table_Store], the column Purchase Loca is fully dependent on the primary key of that table, i.e. Store_id.

III Normal Form

A database is in the III normal form if it satisfies the following conditions —

- it is in the 2nd normal form.
- there is no transitive functional dependency.

By transitive functional dependency we mean — A is functionally dependent on B & B is functionally dependent on C. In this case, C is transitively dependent on A via B.

Eg: Table Book Detail

B_id	G1_id	G1_Type	Price
1	1	Gravel	25
2	2	Sports	14
3	1	Gardening	10
4	3	Travel	12
5	2	Sports	17

Here in the above table, B_id determines G1_id, G1_id determines G1_type & B_id determines G1_type via G1_id, & we have transitive func. dependency. & this structure doesn't satisfies the III normal form.

To bring the table into 3rd normal form, we split it into 2 tables —

Table Book

B_id	G1_id	Price
1	1	25
2	2	14
3	1	10
4	3	12
5	2	17

Table Genre

G1_id	G1_Type
1	Gardening
2	Sports
3	Travel

now, all the non-key attri. are fully functionally dependent on the primary key. i.e. in Table Book, Price is dependent on B_id & G1_id is also dependent on B_id & in Table Genre, G1_type is dependent on G1_id.

BCNF (Boyce & Codd normal form)

- BCNF is one of the forms of the database normalization.
- A database table is in BCNF if & only if there are no trivial functional dependencies of attri. on anything other than a superset of a candidate key.
- BCNF is also sometimes referred to as 3.5NF.
- BCNF was developed by Raymond Boyce & E.F. Codd.
- BCNF is an extension of 3NF and for this reason it is often termed as 3.5NF. 3NF states that all data in a table must depend only on that table's P.K & not on any other field of the table.
- At first glance it would seem that BCNF is same as that of 3NF but in rare cases like when a table with two or more overlapping candidate keys exits.

Eg:

emp-id	emp-nat	emp-dept	dpt-type	dpno
1001	Indian	Prodn & Planning	D001	200
1001	Indian	Stores	D001	250
1002	American	design	D134	100
1002	American	purchase	D134	600

Functional dependencies :

emp-id → emp-nat.
emp-dpt → dpt-type, dp-no.,

Candidate key: emp-id, emp-dpt.

The table is not in BCNF as neither of the emp-id, emp-dpt are alone keys.
To make the table comply with BCNF we can break the table in 3 tables-

emp nationality table:

emp-id	emp-nats
1001	Indian
1002	American.

other columns (usually of a second reln).
The example of inclusion dependency is a foreign key constraint or Referential Integrity constraint as it states that the referring column in one reln must be contained in the primary key column of the refrenced reln.

Table-1

ENO	Name	Age	DNO → Foreign key
1	Ankit	21	10
2	Ram	20	11
3	Raja	19	14 → not allowed as DNO 14 is not defined as p.k. of T-2
4	Rohan	18	10 & in table 1 DNO is foreign key defined.

Table-2

Primary key ← DNO	DLocation
10	Uttarakhand
11	U.P.
12	M.P.

Objectives: To formalize two types of interrational constraints which cannot be expressed using FD's or MVD's —

- Referential Integrity constants.
- class / subclass relationships.

Inclusion dependencies are mostly key-based, i.e. such involve only keys.
Referential integrity constraints are a good example of key-based inclusion dependencies.
ER Diagram that involves ISA hierarchies also leads to key based inclusion dependencies.

Multivalued Dependencies: Multivalued Dependencies are removed using 4th Normal form.

Consider a reln Faculty (FID, Course, Book) which consists of two multivalued attri. (Course & Book). The two multivalued attri. are independent of each other.

FID	Course	Book
1	C1 | C2	B1 | B2
2	C1	B1

It is clear that there are multiple copies of Course & Book. This is an example of a multivalued dependency. It occurs when a reln has more than one independent multivalued attribute.

When a Reln R has attri. A(FID), B(Course), C(Book) such that —

- A determines a set of values for B.
- " " " " " C &
- B & C are independent of each other.

The multivalued dependencies can be indicated as —

- (FID →→ Course).
- (FID →→ Book).

Aggregate funcns: These are used to compute against a "returned column of numeric data." from SELECT statement.

- These funcn perform a calculation on a set of values & a return a single value.
- except for count aggregate funcn ignore NULL values.
- These funcn are frequently used with the GROUP By clause of the SELECT statement.

8200.

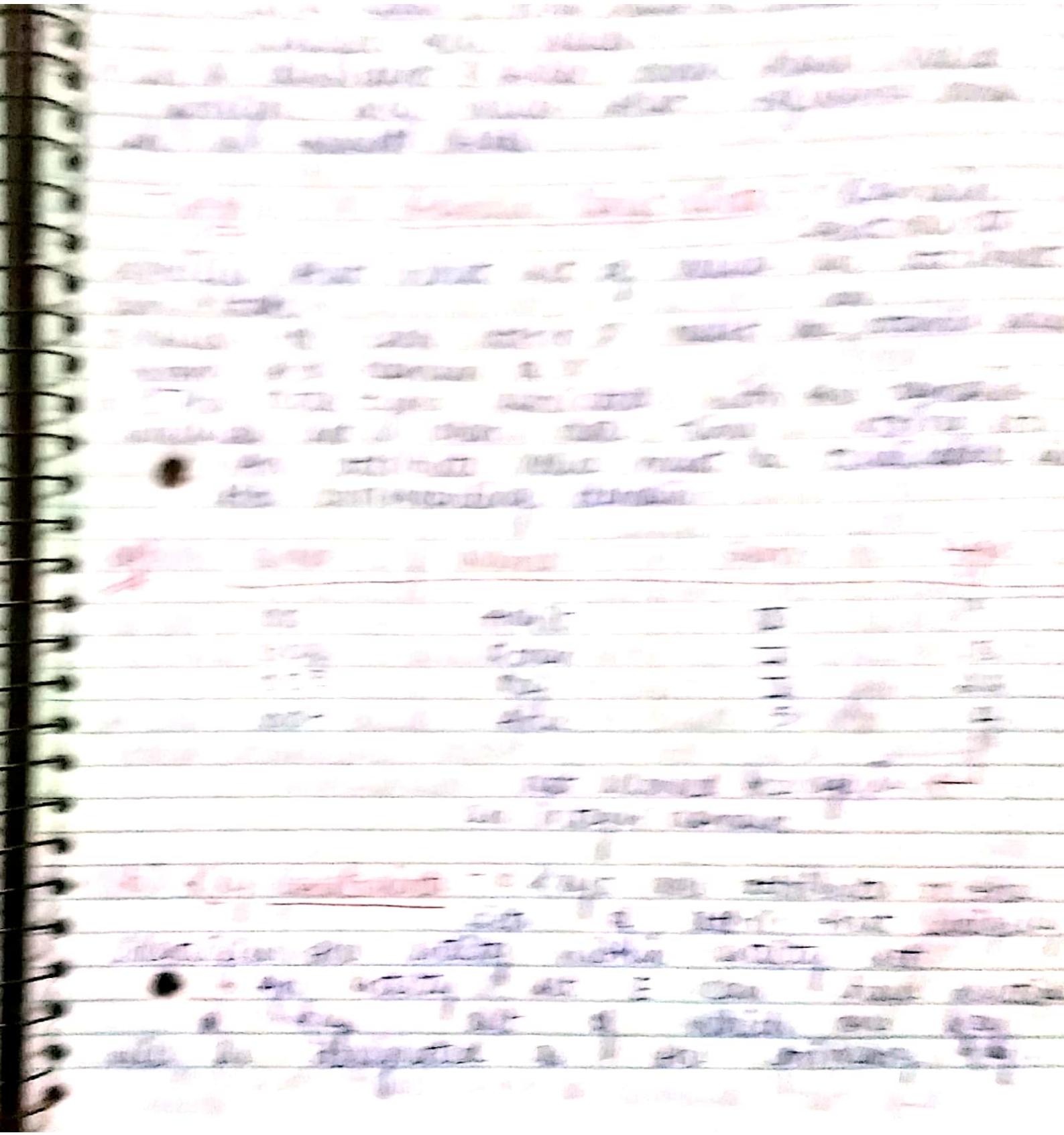
Constraints :

- Constraints are a very important feature in a relational model.
- Relational model supports well defined theory of constraints on attri. or tables.
- Constraints are important because they are used to specify the semantics of data in the data base
- Constraints are the rules to enforce DBMS to check data satisfies the semantics.
- These are the restrictions on the contents of database or on the database operation.

8200.

Constraints :

- Constraints are a very important feature in a relational model.
- Relational model supports well defined theory of constraints on attri. or tables.
- Constraints are important because they are used to specify the semantics of data in the data base
- Constraints are the rules to enforce DBMS to check data satisfies the semantics.
- These are the restrictions on the contents of database or on the database operations.



Need: • Constraints in a database provide a way to guarantee that -
  - the values of individual columns are valid.
  - in a table, rows have a valid PK or a unique key values.
- in a dependent table, rows have valid foreign key values that reference rows in a parent table.

Types: (a) Domain Constraints - • Domain Constraints specifies that what set of values an attribute can take.
- Value of each attrib 'x' must be an atomic value from the domain of 'x'.
- The data types associated with the domains include int, char, date, time, string etc.
- An attribute value must be available in the corresponding domain.

eg:

S.NO	Name	Sem.	Age.
001	Ankit	I	19
002	Rohan	I	18
003	Raj	I	22
004	Anu	5	A

not allowed bcz age is an integer domain

(b) Key Constraints - • Keys are attributes or the set of attri. that uniquely identifies the entity within entity set. E can have multiple

- Primary key must have unique and not null values in the relational table.

eg 8

S.NO	Name	Sem	Age
8001	Ankit	I	19
8002	Raj	I	18
8003	Ram	II	22
8002	Aadi	V	23

↳ not allowed as PK values must be unique.

(C) Integrity Rule I or Entity Integrity Constraints

- Entity integrity is the mechanism that the system provides to maintain primary keys.
- PK serves as a unique identifier for rows in a table.

Entity Integrity ensures two properties for the PK -

- PK is unique for a row.
- PK is not null.

The uniqueness property ensures that the PK of each row uniquely identifies it.

Second property ensures that the PK has meaning; has a value. no compo of the key is missing.

If a relⁿ have a null value in the PK key attr, then uniqueness prop. of the PK can't be maintained.

eg 5

SNO	Name	Sem	Age
001	Ankit	I	19
002	Raj	II	20
003	Ram	III	22
	Roy	IV	25

↳ not allowed; as P.key can't be a Null value.

R) Integrity Rule 2 - • Integrity Rule 2 is also called as the Refrential Integrity Constant.

- This rule states that if the foreign key in Table 1 refers to the primary key of Table 2, then every value of the foreign key in Table 1 must be null or be available in Table 2.

eg 6

ENO	Name	Age	Dno
1	Ankit	19	10
2	Raj	22	11
3	Ram	23	14
4	Roy	24	10

Dno → Foreign key.

PK.

Dno	DLocⁿ
10	UP
11	UP
12	AP

not allowed as Dno 14 is not defined as PK in Table 2. & in Table 1, Dno is a foreign key.

- Refrential Integrity is the mechanism the system provides to maintain the foreign key.
- Refrential integrity ensures only one property for foreign key -

the valid foreign key value must always refer to an existing PK or contain a null.

The defⁿ of a FK must specify the table whose P.K. is refered.

Let the table in wch the foreign key is defined be the foreign key i.e. 'Table 1' and the table that defines the P.K. is the Table two / master table.

Table if the corresponding records in the master table donot exists.

- Records of the master table can't be alted or updated if corresponding records in the detail table actually exists.

Relational Algebra: RDBSystem are expected to be equipped with a query lang. that can assists its users to query the instances.
2 — Reln Algebra & Reln calculus.

- Reln Algebra is a procedural query lang. which takes instances of reln as an input & gives instances of reln as an output.
- It uses operators to perform query. Operators can be either unary or binary.
- RA is performed recursively on a reln & the intermediate results are also considered as reln.
- They accept reln as their inputs & yields reln as their output.

The fundamental opern of the RA are—

- **Select opern:** Select opern is the unary opern. It requires only one operand to operate.
- The operand is nothing but a reln.
- When this opern is applied on the reln the resulting reln contains the filtered tupples.
- The filtern depends on the condn for the select opern.
- The tupple which satisfies the condn are given as a resulting reln.
- The select opern is symbolically rep. by — σ
- syntax — $\sigma_{condition}(R)$.

The relative operators used to frame a condn are — >, <, >=, <=, =, <>, LIKE, BETWEEN -- etc.

eg: Let us consider a Reln R (A, B, C, D) with their respective tupple values-

A	B	C	D
100	16	Pilani	18000
101	15	Pilani	23000
102	16	ND	12000
104	15	Hyderabad	3500
106	12	Bengaluru	6500
107	9	"	4000
108	8	Hydera.	18500
109	7	Bhopal	29000
115	5	Agra	32000
118	10	Baroda	13400

$\sigma_{B >= 10}(R)$

A	B	C	D
100	10	Pi	
101	15	"	
102	16	ND	
104	15	H	
106	12	B	
118	10	Baroda	

Projecn: When this opern is applied to the reln, then the resulting reln contains all the tupples with selected attrib.

- The attri. to be selected are specified along with projecn operator.
- The projecn operator is symbolically represented as — π.

$\pi_{A1, A2 \ldots An}(R)$ — syntax.

This is the simplest operⁿ that simply the list of attri. are selected & placed in the order of interest.

- The resulting relⁿ contains all the tupples with related values of selected attrib.

eg: From book

Union: It performs binary union b/w the '2' given relⁿ.

- Its notatⁿ is – $r \cup s$.

where r & s are either database relⁿ or result set.

- for a union operⁿ to be valid, the following condⁿ must hold –
  1) r & s must have the same no. of attributes.
  2) attri. domains must be compatible.
  3) duplicate tupples are automatically eliminated.

eg: S_1

Sid	Sname	Rating	Age
22	Ram	7	21
31	Raj	8	22
58	Roy	10	23

S_2

Sid	Sname	Rating	Age
23	Reet	12	14
24	Mannu	13	24
31	Raj	8	22
58	Roy	10	23

$S_1 \cup S_2$

Sid	Sname	Rating	Age
22	Ram	7	21
31	Raj	8	22
58	Roy	10	23
23	Reet	12	14
24	Mannu	13	24

Intersection: Denoted as – $R \cap S$.

- returns a relⁿ instance containing all the tupples that occur in both R & S.
- R & S must be union compatible.
- Schema of the result is that of R.

eg: Considering the above eg:

$R \cap S$ →

Sid	Sname	Rating	Age
31	Raj	8	22
58	Roy	10	23

Difference: denoted as $R-S$.

- returns a relⁿ instance containing all the tupples that occur in R but not in S.
- Schema of the result is the schema of R.

eg: As above eg:

$S_1 - S_2$ →

Sid	Sname	Rating	Age
22	Ram	7	21

- **Relational Calculus:** It is a non-procedural Query Lang. It uses mathematical predicate calculus instead of algebra.

get the [illegible] the descriptn about the query to get the result & the RA provides gives method to get the result.

- It informs the system what to do but with the rel^n but does not inform how to perform it.

eg6 steps involving in the listing of all the students who attended the database course in RA would be -

- SELECT the tuples from COURSE rel^n with COURSE_NAME = 'DATABASE'
- PROJECT the course COURSE_ID from above result.
- SELECT the tuples from STUDENT rel^n with COURSE_ID resulted above.

But in the case of Relational Calculus, it is defined as -
Get all the details of the students such that each student have course as 'Database'
Rel^n Calculus tells what needs to be done to get the students with 'database' course.
But it does not tells us how we need to proceed to achieve this.
Rel^n Calculus is a explanative way to tell the query -
There are 2 types -
Tuple Rel^n Calculus (TRC)
Domain " " (DRC).

- A Tuple Rel^n Calculus is non procedural Query Lang. which specifies to select the tupples in a rel^n.
- It can select the tuples with range of values or tuples for certain attribute values etc.
- The resulting rel^n can have one or more tuples.
  - Denoted as -
    $\{t \mid \text{condit}^n(t)\}$ -

    t = resulting tuples.

eg^n $\{t \mid EMPLOYEE(t) \text{ and } t.SALARY > 10000\}$ -

implies that it selects the tuples from EMPLOYEE rel^n such that resulting rel^n we have salary > 10000.

$\{t \mid EMPLOYEE(t) \text{ and } t.DPT\_id = 10\}$
selects all the tuples of employee name who work for Dpt 10

Here, $t.salary$ & $t.Dpt$ are the tuple variable.
in Ist ex. we have specified condit^n. To display those employee salary > 10000, here salary is called as the bound variable. Bound variables are those whose meaning will not change if the tuple variable is replaced by another tuple variable.

in second eq = DPT_ID = 10, this is called as the Free variable. i.e. if we change DPT_ID in this condit^n its same the vocabulary, EMP_ID, the meaning of the query changes.

All the conditions used in the tuple exp. are well formed formula.

Domain Relational Calculus uses list of attributes to be selected from the rel^n based on the condition.
It is same as TRC but it differs in selecting the attributes rather than selecting whole tuples.

Denoted as: $\{\langle a_1, a_2 \dots a_n \rangle \mid P(a_1, a_2 \dots a_n)\}$

where $a_1, a_2 \dots a_n$ are the attri. of the rel^n & P is the condit^n.

eg: Select EMP_id and EMP_Name of employees who work for DPT = 10

$\{\langle EMP\_ID, EMP\_NAME \rangle \mid \langle EMP\_ID, EMP\_NAME \rangle \in EMPLOYEE \wedge DPT\_ID = 10\}$

Get name of the department that Alex works.

$\{DPT\_NAME \mid \langle DPT\_NAME \rangle \in DPT \wedge \{DPT\_ID\} \langle DPT\_ID \rangle \in EMPLOYEE \wedge EMP\_NAME = Alex\}$

Transcn: ° A Transcn can be defined as a group of tasks.
° A single task is the minimum processing unit which can't be divided further.
eg: ° Suppose a bank employee transfers Rs. 500 from A's account to B's account. This very simple & small transcn involves several low-level tasks.
° A Transcn is a very small unit of a program & which can't be further. It may contain several low level tasks.
° A transcn in a database system must maintain Atomicity, Consistency, Isolation & Durability, commonly known as ACID properties. - in order to ensure accuracy, completeness & integrity of data.

Atomicity - ° This property states that a Transcn must be treated as a atomic unit i.e. either all of its opern are executed or none.
° There must be no state in a database that is left partially completed.
° States should be defined either before the execn of the transacn or after the execn/ abortn/ failure of the transacn.

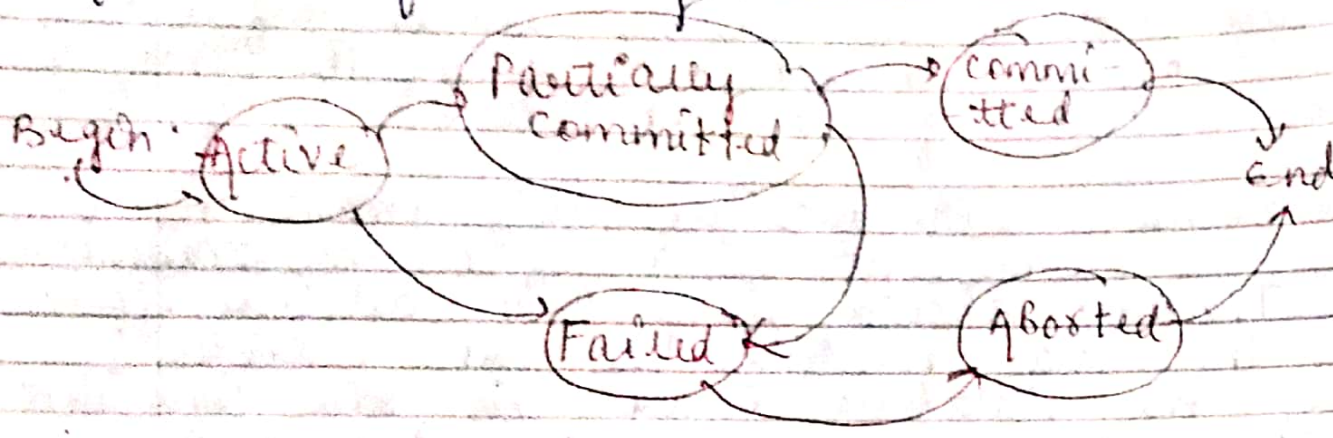
Consistency - ° The database will remain in a consistent state after any transacn.
° NO Transacn should have any adverse effect on the data residing in the database.
° If the database was in the consistent state before the execn of transacn, it must remain consistent after the execn of the transcn as well.

Isolation - ° In a database system where more than one transcn are being executed simultaneously & in I/r, the property of isol^n state that all the transcn will be carried out & executed as if it is the only transcn in the system. No transcn will affect the existence of any other transacn.

Durability - ° The database should be durable enough to hold all its latest updates even if the system starts or fails.
° If a transcn updates a chunk of data in a database & commits, then the database will hold the modified data.
° If a transacn commits but the system

fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.

State of Transaction: A transaction in a database can be in either of the following states -



Active - In this state, the transaction is being executed, this is the initial state of every transaction.

Partially Committed - When a transaction executes its final operation, it is in the partially committed state.

Failed - A transaction fails, if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.

Aborted - If any of the checks fails and the transaction has reached its failed state, then the recovery manager rolls back all its write operation on the database to bring it to its original state where it was prior to the execution of the transaction.

Open
Loop.
Fetch data from cursor.
Exit Loop.
Close

Hie Model	Network model	Rel^n Model
◦ one to many. rel^n & one to one rel^n.	◦ many to many.	◦ one to one. n to many many to n.
◦ Based on Parent child rel^n ship	◦ may have. many parents as well as many children	◦ Based on rel^n data structure.

Hie Model	Network model	Rel^n Model
◦ one to many rel^n & one to one rel^n.	◦ many to many.	◦ one to one. n to many many to n.
◦ Based on Parent child rel^n ship	◦ may have many parents as well as many children	◦ Based on rel^n data structure.

◦ does not provide an independent stand alone structure	◦ uses CODASYL.	◦ it uses powerful oper^n such as SQL lang.
◦ algo. are complex and symmetric	◦ same	◦ simple and symmetric
◦ we can't insert any info^n of a child if it doesn't have any parent	◦ doesn't suffer from insert anomaly.	◦ same.
◦ multiple occurens of child records with lead to problems of inconsistency during update oper^n	◦ free from update anomaly	◦ same.
◦ Del^n of parents results in del^n of child records.	◦ free from delete anomalies	◦ free.

Primary key : is a candidate key that is most appropriate to be the main ref. of the table.

eg: STD ID.

⇒ P.K. are mandatory for each table and record values for its primary key.

• While choosing from the pool of candidate key always choose a singe simple key over composite key.

→ Candidate Key is the subset of the super key.

A candidate key is a single field or the combination of the fields that uniquely identifies each record in the table.

Least combinatn of fields distinguishes candidate key from super key.

Every table must have one cand. key but at the same time can have several.

eg. sid F_name L_astname Course

Candidate key

sid uniquely identifies the student in a table.

To be candidate keys-

- must have unique values.
- " not " null ".
- mini. no. of fields.
- uniquely identifies.

Foreign Key is generally a P.K. from one table that appears as a field in another where one table is linked with another.

if a table A with P.K. X then X is a field of B.

eg. sid Fname Lname Courid.

relh.

cid C-Name. foreign key.

Advantages of E-R

- Straightforward ~~etc~~ reln rep.–

having designed ~~for~~ E-R model for database app. the reln rep. of database model become rel.

- easy conversion of E-R diag to another model – (net / hier)

- Graphical rep. for better understanding – diagrammatical rep. of various entities, th ~~sets~~, attrs. & reln b/w entts. helps in clear understanding of data struct. and minimize redundancy.

Degree: no. of participatn entities in a relnship defines degree of a relnshp.

Binary = degree 2
Ternary = 3
n-ary = n.

Mapping Card.: no. of entities in one entity set wh. can be associated with the no. of entities of other set via relnshp set.

to any malicious users.

- Interpret back up & recovery.
- monitoring performance. and responding changes if req.