2017-18.

1(a). Relation →

A relation is a table that consists of rows and columns, where each row represents a record and each column represents a specific attribute a field.

Cardinality →

Cardinality refers to the number of tuples a rows present in the relation or table.

Or

Degree → Degree refers to no. of attributes or columns in a relation.

For the given relation, Student,

Cardinality = 100.

Degree = 7

(b) Super Key → A superkey is a set of one or more attributes which can uniquely identify a row in a table.

Or

Candidate Key → The minimal set of attributes that can uniquely identify a record in a table is called a candidate key.

Primary Key → The candidate key chosen to uniquely identify each record in a table is called a primary key.

A relation has only 1 primary key. A primary key cannot contain NULL values.

Foreign key → A foreign key is an attribute or set of attributes whose value is derived from the primary key of some other table or the same table. It may or may not be unique in the referencing relation.

For a relation with n attributes & K candidate key.

No. of superkeys = $k2^{n-1} - 2^{n-k}$.

e.g. consider the relation

Student { Rno, Addmission_no, Name, Address, Mob_no, Course }.

Candidate keys ⇒ Rno, Addmission_no.

Primary key ⇒ Rno.

Super keys ⇒ $2.2^{6-1} - 2^{6-2}$

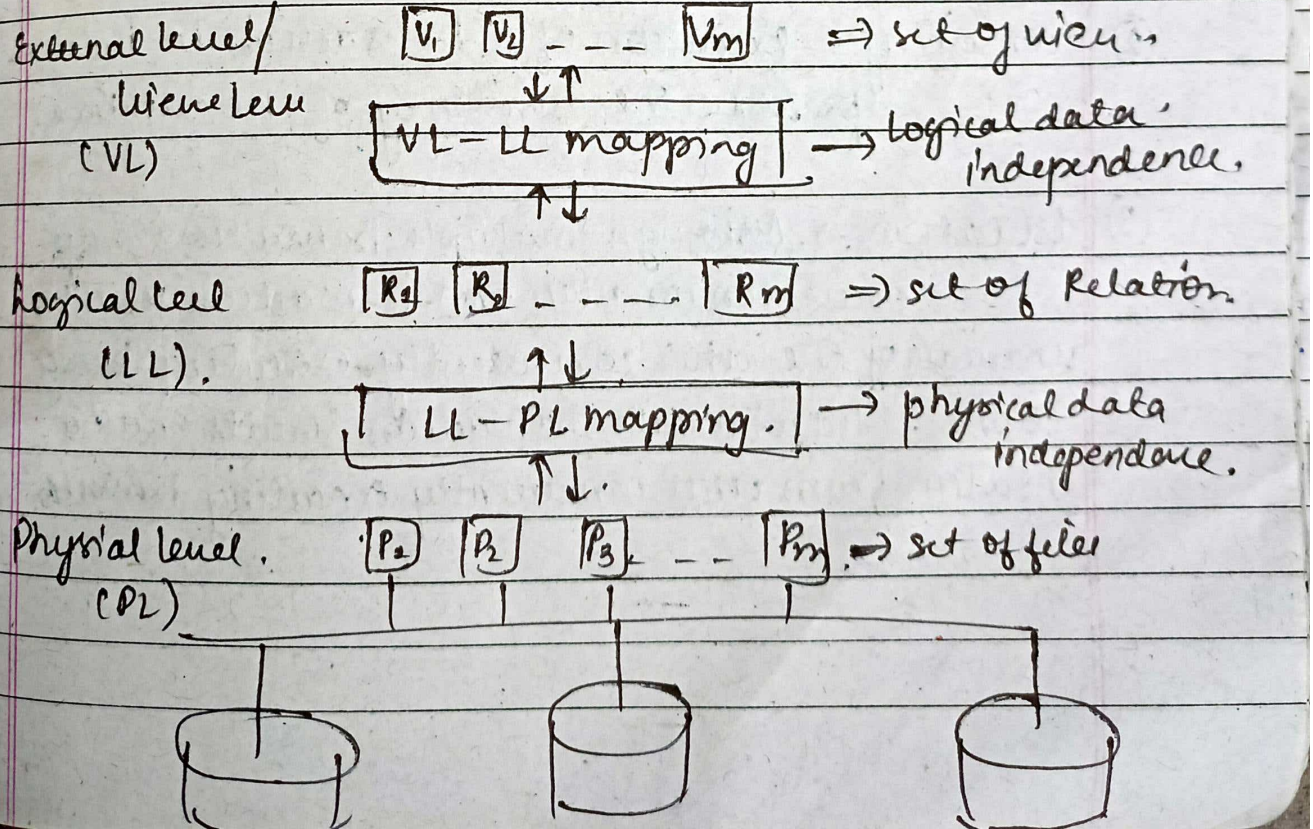
$= 2.2^5 - 2^4$.

$= 64 - 16$.

$= 48$.

(1). Logical Data Independendence.
→ It refers to the characteristics of being able to change the conceptual schema without having to change the external schema.
→ It is used to separate the external level from the conceptual view.
→ It occurs at the user interface level.
→ VL-LL mapping happens.

Physical Data Independence.
→ ~~Physi~~ It can defined as the capacity to change the internal schema ~~has~~ without having to change the conceptual schema.
→ e.g. If we do any changes in the storage size of the database server, then the conceptual structure will not be affected.
→ It occurs at the logical interface level.
→ LL-PL mapping occur.



Achieving logical data independence is more difficult as compared to physical data independence since application programs are heavily dependent on the data logical format of the data they access, hence a change in conceptual level might require the change of the entire program application.

(d) Transaction → A transaction is a set of logically related operations. A transaction includes one or more database access operations these can include insertion, deletion, modification or retrieval operations.

ACID Properties →.

① Atomicity →. either all operations of the transaction are reflected or none.

② Consistency →. Execution of a transaction in isolation preserves the consistency of the database.

③ Isolation →. Although multiple transactions may occur concurrently, each transaction must be unaware of the other concurrently executing transactions. Intermediate transaction results must be hidden from other concurrently executing transactions.

(d). Durability → After a transaction is completed successfully, the changes made to the database must persist, even if there are system failures.

e). Need of Normalization →.

It is important that a database is normalised to minimise redundancy (duplicate data) and to eliminate various anomalies such as updation, insertion of deletion. Normalisation ensures that the design of database is efficient, organized & free from anomalies.

Lossless & Dependency Preserving Decompositions

A decomposition is called lossless decomposition. If a relation R is decomposed into relations R_1 and R_2 such that

(i) $R_1 \cup R_2 = R$ (ii) $R_1 \cap R_2 \neq \phi$ (iii) $R_1 \cap R_2$ = superkey of R_1 or R_2

In a dependency preserving decomposition, at least one decomposed table must satisfy every dependency. If R is decomposed into R_1 and R_2, dependencies of R must be a part of R_1 or R_2 or derivable from combination of FDs of R_1 and R_2.

eg. R (A, B, C) be a relation with FD (A → B).

R_1 (AB) R_2 (AD)

⇒ It is a lossless & dependency preserving.

Now $R_1 \cup R_2$ = ABCD. & decomposition

$R_1 \cap R_2$ = A which is super key of R_1.

(4). $R(ABCDE)$ → $R_1(ABC)$, $R_3(CDE)$, $R_2(BCD)$

$FD_s\{A \to B, C \to D, A \to E\}$.

$R_1(A, B, C)$. $R_3(C, D, E)$. $R_2(B, C, D)$.

$FD \Rightarrow A \to B$ $FD \Rightarrow C \to D$. $FD \Rightarrow B \to D$.

$R_1 \cap R_2 = BC$. ⇒ superkey of R_1

$R_2 \cap R_3 = CD$.

(q) Union Compatibility →

Union compatibility of relation implies that the participating relations must fulfil the following condition.

→ Same degree ie. same no. of attributes.

→ Same domain of each corresponding pair of attribute of relation A & relation B ie.

Dom(A) = Dom(B)

Union operation → It returns all tuples which are present either in A or B.

Intersection Operation ⇒ All tuples common to both A as well as B.

Difference Operation → (A − B) ⇒ Tuples in A but not in B.

These operations require union compatibility because all these operations are binary set operations. The tuples of the relations are directly compared under these operations. Hence the tuples should have same no of attributes with the same domain.

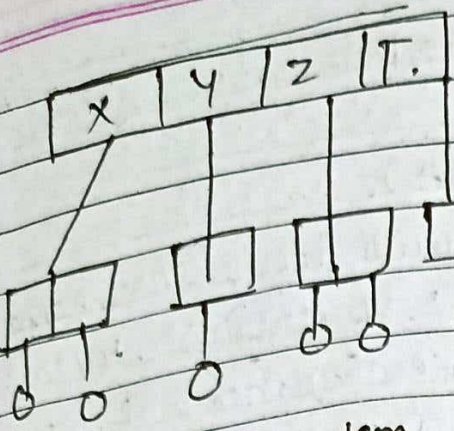
2. (a) DBMS ⇒

A DBMS is a collection of interrelated data and a set of programs to access those data. The primary goal of a DBMS is to provide a way to store & retrieve database information. The major purpose of a database system is to provide users with an abstract view of data. It hides certain details of how data is stored & maintained.

Difference b/w file system & DBMS.

→ The file system is basically a way of arranging the files in a storage medium like a hard disk. It consists of different files which are grouped into directories. The directories further contain other folders & files. The file system performs basic operations like management, file naming, giving access rules etc.

→ A DBMS stores the data in the form of relations. Ie. table. In DBMS, the data can be fetched using SQL queries & relational algebra.

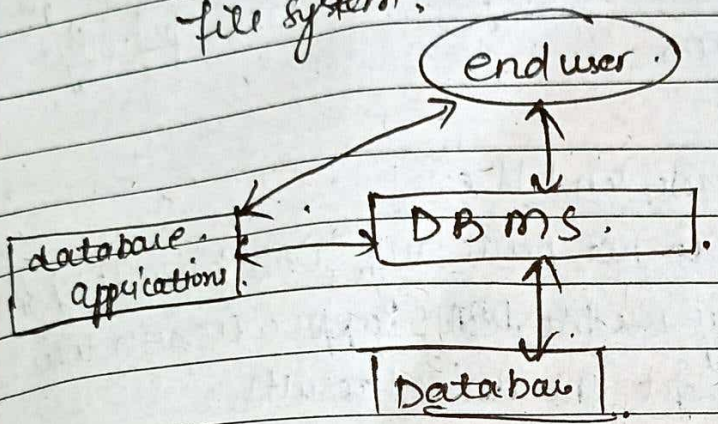


master file director.

user file directory.

files.

file system.



DBMS.

Advantages of DBMS.

- DBMS reduces data redundancy.
- It provides inhouse tools for data backup & recovery. unlike file system.
- There is more data consistency because of the process of normalization.
- More security Mechanisms.
- Multiple users can access data at a time.

Disadvantages →.

- Comparitively higher cost than a file system.
- More complexity in handling as compared to filesystem.
- The use has to write procedures for managing database.

(b). The different types of database end users are-

(1) Database Administrator (DBA).

DBA is a person or team who defines the schema and also controls the 3 levels of database. He/she is responsible for providing security mechanism for the DBMS & ~~allows~~ manages the various privilages to different users.

(2) Naive/ Parametric end Users.

These users do not have any DBMS knowledge but they frequently use the DBMS applications in their daily life to get the desired results.

(3) Database Designers.

These are the users who design the structure of database. It is the responsibility of database designers to understand the requirements of different user groups and then create a design which satisfies the need of all the user groups.

(4) Application Programmers—.

They are the back end programmers who write code for the application programs. They are the computer proffessionals.

(5) System Analyst →.
They are users who analyse the requirements of parametric end users they check if all requirements of end users are met.

(6) Specialized users.
These are sophisticated users who write specialized database applications that does not fit into traditional data-processing framework.

3(a). Closure of an attribute →.
It can be defined as the set of attributes which can be functionally determined from an attribute.

Algorithm to find closure of an attribute set.
→ Add elements of attribute set to the result set.
→ Recursively add elements to the result set which can be functionally determined from the elements of the result set.

PTO

(A, B, C, D)

AB → C

AB⁺ = {A, B, C}

AB D⁺ = {A, B, C, D}

R(ABCDEH).

F = {A → E, B → D, C → ADE, BD → H}.

X = BCD

Now X^+ = BCD$^+$ = {B, C, D, A, E, H}.

(b) Concurrency ⇒

Concurrency is the execution of multiple transactions at the same time.

Concurrency control is needed in DBMS to allow multiple transactions to run concurrently without impacting integrity. It helps to prevent data corruption & inconsistencies.

Without concurrency control, concurrent transactions can lead to problems like lost update, inconsistent reads & other data anomalies.

e.g. Consider a banking system where two users simultaneously attempt to withdraw money from the same account. Without concurrency control mechanism in place, both transaction might read the same initial account balance, perform their calculations independently & update balance without considering each other's changes. This could result in incorrect final balance or lost update as one transaction's change may overwrite others.

2NF =>
→ Every cell should contain atomic values.
→ There should be no partial dependency.
ie. AB → C ⇒ A ↛ C or B ↛ C

3NF ⇒
→ The relation should be in 2NF.
⇒ There should not be a transitive dependency ie.
a non-prime attribute should not determine non-prime attribute.

BCNF =>
- It is a stricter form of 3NF.
→ for any A → B, A should be a super key.
ie. non-prime cannot determine prime.

R ⇒ Lots (property#, country-name, Lot# Area, price, Tax-rate)

property #$^+$ = { country-name, Lot#, Area, Price, Tax-rate, property# }

country-name, Lot# = { Property#, Area, Price, Tax-rate, country-name, Lot# }

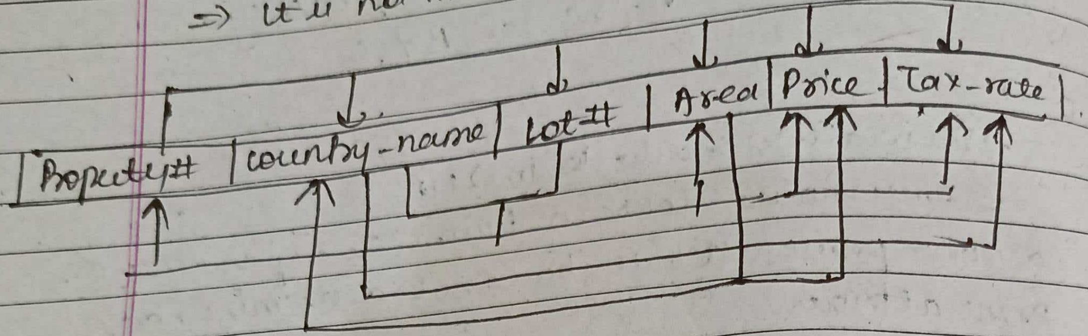
Area$^+$ = { Area, price, country-name, tax-rate }.

⇒ Candidate keys ⇒ { property #, { country-name, Lot#) }.

Prime Attributes ⇒ property#, country-name, Lot#

Country-name , Lot# → Tax-rate .
country-name → Tax-rate .

⇒ It is not in 2NF.



for 3NF, the table can be decomposed a

R1
(property#, country-name, Lot#, Area, price).

R2
(country-name, tax-rate)

country-na → ~~tax~~-ra.

→ pr.

③ Now →.

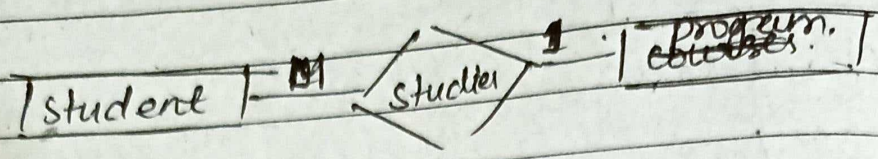
3NF

property#, country-na, Lot#, Area.

Area, Price

Property#, country-na, Lot#

~~Country~~, na.
Area, country na.

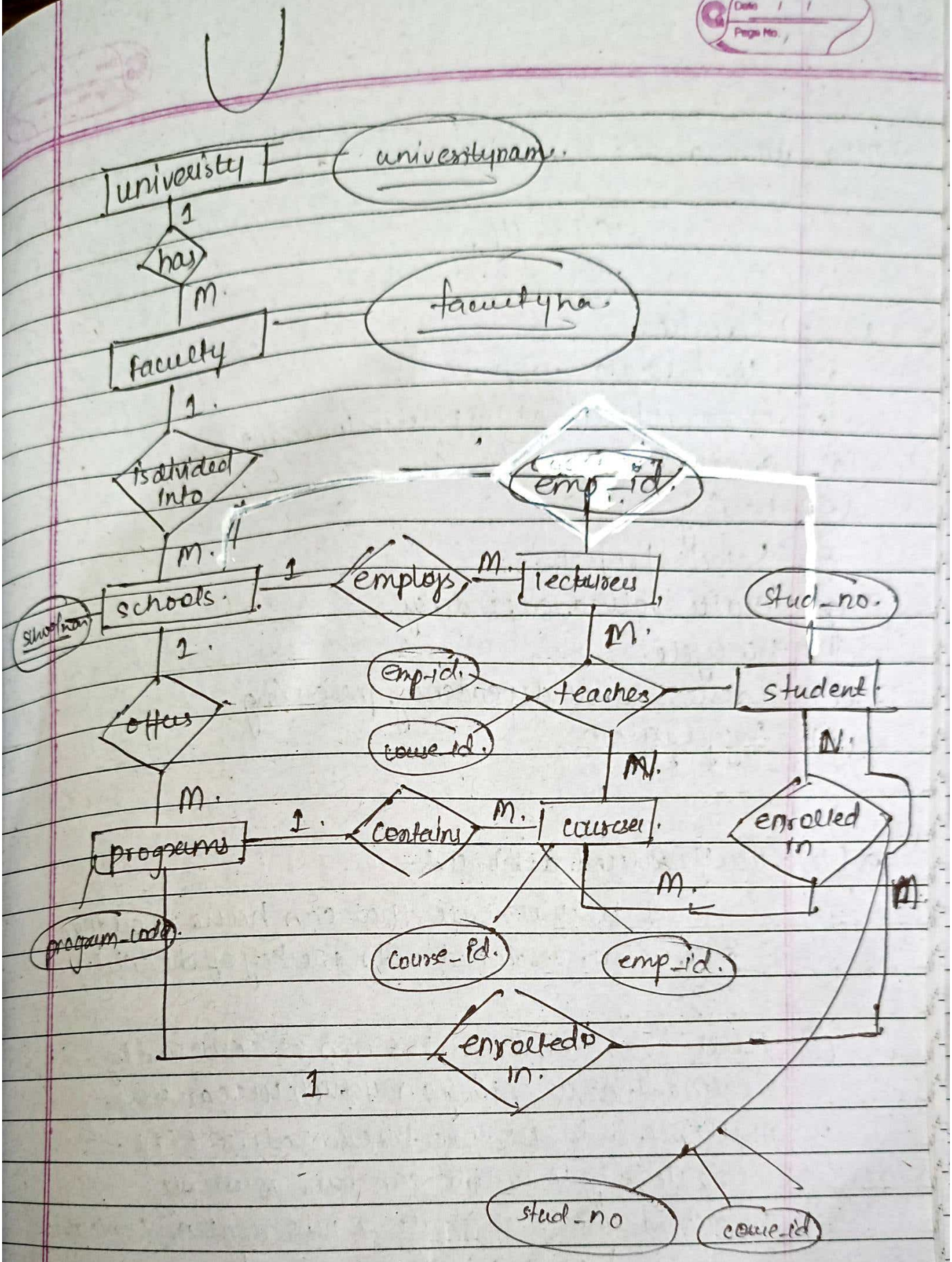


The entities are -

1. University.
2. Faculty.
3. School.
4. Program.
5. Course.
6. Lecturer.
7. Student.

Finding Relationship.

	University	Faculty	School	Program	Course	Lecturer	Student
University		has.					
faculty			divided into.				
School.				offers.		employs.	
Program.					contains.		
course.							
Lecturer.					teaches.		
Student				enrolled	enrolled		


university
universityname
1
has
M
faculty
facultyname
1
is divided into
M
schools
schoolname
1
offers
M
programs
program_code
1
contains
M
courses
course_id
emp_id
M
employs
1
M
lectures
emp_id
M
teaches
course_id
M
Stud_no
student
M
enrolled in
M
1
enrolled in
Course_id
emp_id
stud_no
course_id

The Relation can be.

2013-14.

1. (a) degree.
(b) logical data independence.
(c) referential integrity constraint.
(d) shadow.
(e) having.
(f) Select, Project.
(g) Multi-valued dependency.
(h) no cycle.
(i) Lossless and dependency preserving.
(j) Aggregation.

2(b) Multivalued attributes -

It is an attribute that can have more than one value associated with the key of the entity.

(i) Multi-valued attributes can be resolved by creating a new entity, in which we can store multiple instances, one for each value of the attribute. e.g Astudent can have multiple contact numbers so create a new relation contact_info as.

Student_id	Contact_no
123	9956xxxxxx
123	98836xxxxx

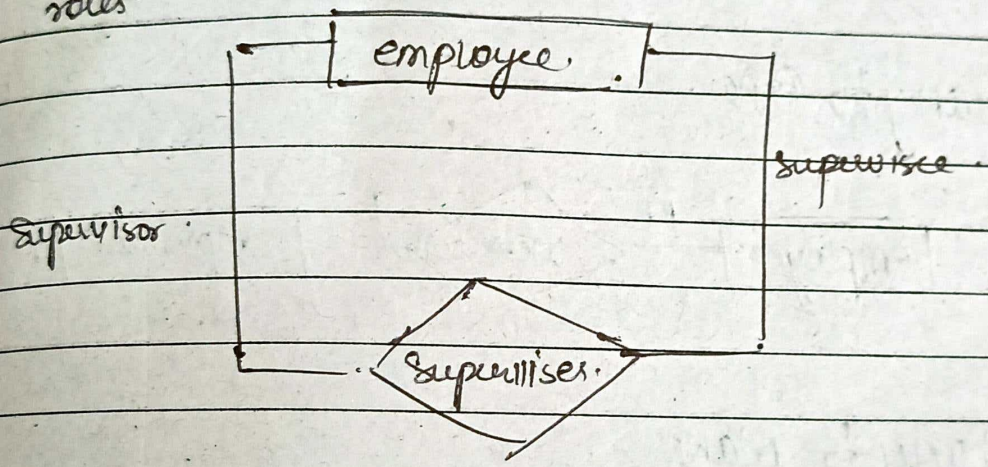
(2) Another approach can be to add as many columns for that attribute as there are values for the multivalued attribute.

e.g.

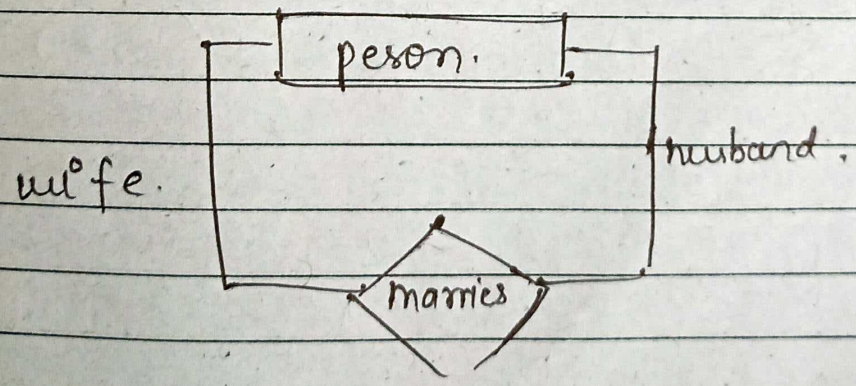
Studid	Contact_no1	contactno_2
123	9456xxxx	98836xxxx

~~however this approach can~~

2(c). A recursive relation is one in which one entity may participate more than once in a relationship in different roles.



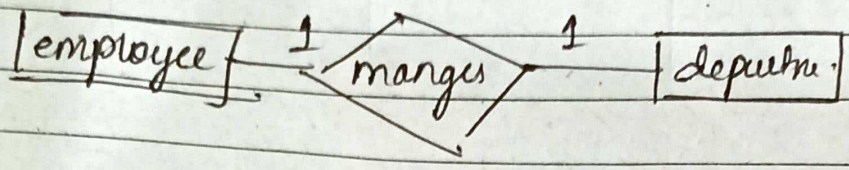
OR



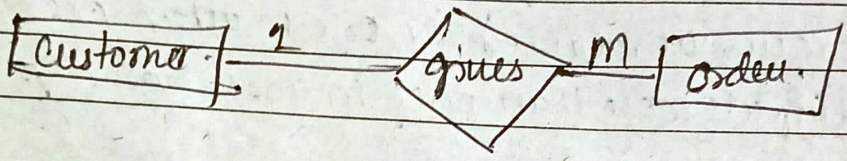
(d) Cardinality Ratio. of an entity set.

The no. of times an entity participates in an relationship set is known as cardinality. It can be of many types -

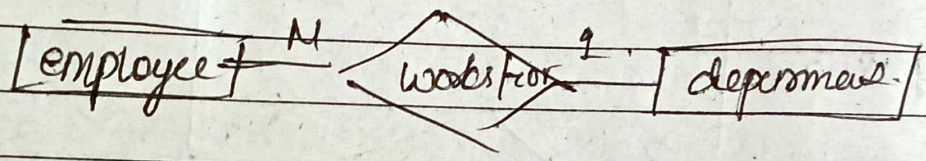
(1) One-to One.



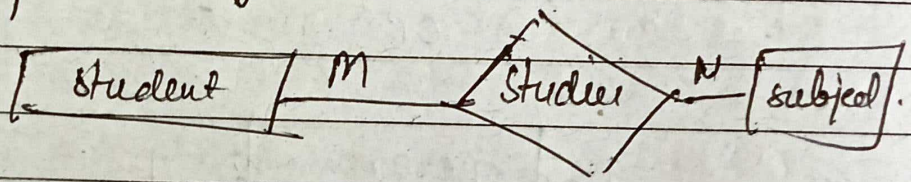
(2) One-to Many.



(3) Many to One.



(4) Many to Many.



8(a). A decomposition of a table R into R_1 and R_2 is said to be lossless iff -

(1) $R_1 \cup R_2 = R$.

(2) $R_1 \cap R_2 \neq \Phi$.

(3) $R_1 \cap R_2$ = superkey of R_1 or R_2.

$R(A, B, C, D, E)$ $R_1 = (A, B, C)$ $R_2 = (A, D, E)$.

$R_1 \cup R_2 = (A, B, C, D, E)$

$R_1 \cap R_2 = A$

Now $A \to BC$, $CD \to E$, $B \to D$, $E \to A$.

$A+ = \{A, B, C, D, E\}$.

for Relation R_1.

$A+ = \{A, B, C\}$. $\Rightarrow$ A is superkey of R_1.

$\Rightarrow$ OR The given decomposition is lossless.

(b). Full functional Dependency →.

An attribute is fully functional dependent on another attribute, if it is functionally dependent on that attribute and not on any of its proper subset.

Let $AB \to C$.

then, $A \not\to C$ and $B \not\to C$.

Partial functional Dependency –

Partial functional depency occurs when a non-prime attribute is fully functional dependent on a part of a candidate key.

Transitive dependency →

A transitive functional dependency occurs when a non-prime attribute determines another non-prime attribute through a chain of dependencies.

i.e. if $P \to Q$ and $Q \to R$.

$\Rightarrow P \to R$ is transitive F.D.

The third normal formal (3NF) is based on eliminating transitive functional dependencies in a relation. A relation is in 3NF if it is in 2NF and there are no transitive dependencies.

Consider the relation $R(A, B, C, D, E)$ with FDs $A \to BC$, $CD \to E$, $B \to D$, $E \to A$.

$A^+ = \{A, B, C, D, E\}$.

$B^+ = \{B, D\}$.

$BC^+ = \{B, C, D, E, A\}$.

$BD^+ = \{B, D\}$.

$BE^+ = \{B, E, A, C, D\}$.

$C^+ = \{C\}$.

$CD^+ = \{C, D, E, A, B\}$.

$E^+ = \{E, A, B, C, D\}$.

$D^+ = \{D\}$.

$\Rightarrow$ Candidate keys are $\to A, BC, CD, E, ED$.

There are no transitive FDs and hence this relation is in 3NF.

Q4(a) the immediate
if transaction
is still alive
modified imme
failure, the
need to be reco

in deffered
does not mod

The immedi

- Undo(T_i).
- Redo (T_i)

(b) Two
Thi
Lock
- C

Q4(a) The immediate database modification technique occurs if transaction modification occurs while the transaction is still alive. In this technique, the database modifications is modified immediately after every operation. In case of a failure, the changes made by the incomplete transaction might need to be recovered.

In deffered database recovery technique, the transaction does not modify the database until it has committed.

The immediate ~~data~~ recovery scheme uses 2 procedures -

- Undo(T_i) → Restores the value of all data items updated by transaction T_i to the old value.

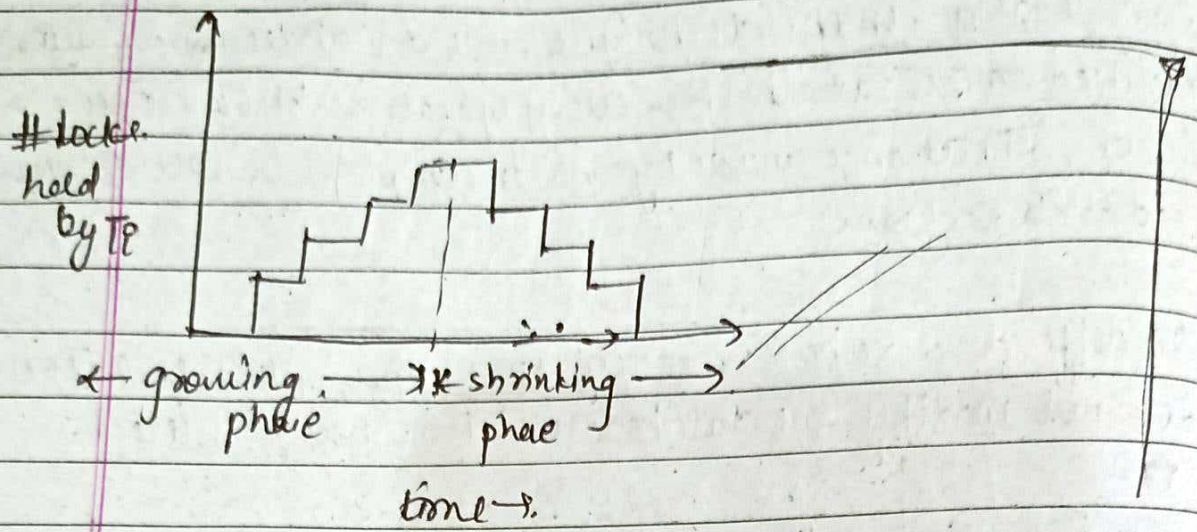
- Redo(T_i) → sets the value of all all data items updated by Transaction T_i to the new value.

(b) Two phase locking Protocol

This protocol requires that each transaction issues lock & unlock requests in two phase.

- Growing phase –
  A transaction may obtain locks, but may not release any locks.
- Shrinking phase –
  A transaction may release locks, but may not acquire any new locks.

Two phase locking protocol ensures confict seriali zability



The two variations of 2PL protocol are —

- Strict 2PL protocol.
  This protocol requires that two phase protocol with the condition that all exclusive-mode locks taken by transaction be held until transaction commits.

- Rigorous 2PL protocol.
  This protocol follows 2PL as well that all aquired locks be held until the transaction is committed.

Strict 2PL protocol is often preffed. because it ensures that any data written by an uncommitted transaction are locked in exclusive mode until the transaction commits, preventing any other transaction from reading the uncommitted data. It also avoids the problem of cascading rollback.

(c) Recoverable Schedule →.

A schedule is recoverable if it allows for the recovery of one database to a consistent state after a transaction failure. Schedules in which transactions that has updated a database must commit before any other transaction reads or writes the same data is called a recoverable schedule.

Recoverable schedules are desirable because failure of a transaction might otherwise bring the system into an irreversibly inconsistent state.

5(a). Natural Join ⇒ It joins combines tuples from two relations that are equal on their common attribute names.

Equi Join ⇒ It combines tuples from two relations based on matched data as per equality condition.

	Natural Join	Equi Join.
Column Name	Tables must have atleast one same column name.	Common column name can be same or different.
~~Data Type~~.		
output.	Only unique columns without repetition.	Both tables in output without missing any column.

Example.

Consider table.

Natural Join.

Stud_id	Name	C_id
01	Anup.	107
02.	Ajay.	106.
03.	Priya.	107

Student

C_id	C_name
105	OS.
106.	DBMS
107	DSA

Course.

Select * from student NaturalJoin Course.

Output →.

Stud_id	Name	C_id	C_name
01	Anup.	107	DSA.
02	Ajay.	106.	DBMS
03.	Priya	107	DSA

The same query using Equi Join is —.

Select * from student Join course ON.
Student.C_id = Course.C_id.

Output

Student.stud_id	Student. Name	Student. C_id	Course. C_id	Course. C_name
01	Anup	107	107.	DSA
02	Ajay.	106	106.	DBMS
03	Priya	107.	107	DSA.

ANSI-SPARC 3 level Architecture.

Q.6(a) The three level architecture aims to seperate each user's view of the database from the way the database is physically represented.

(1) External level / View level.

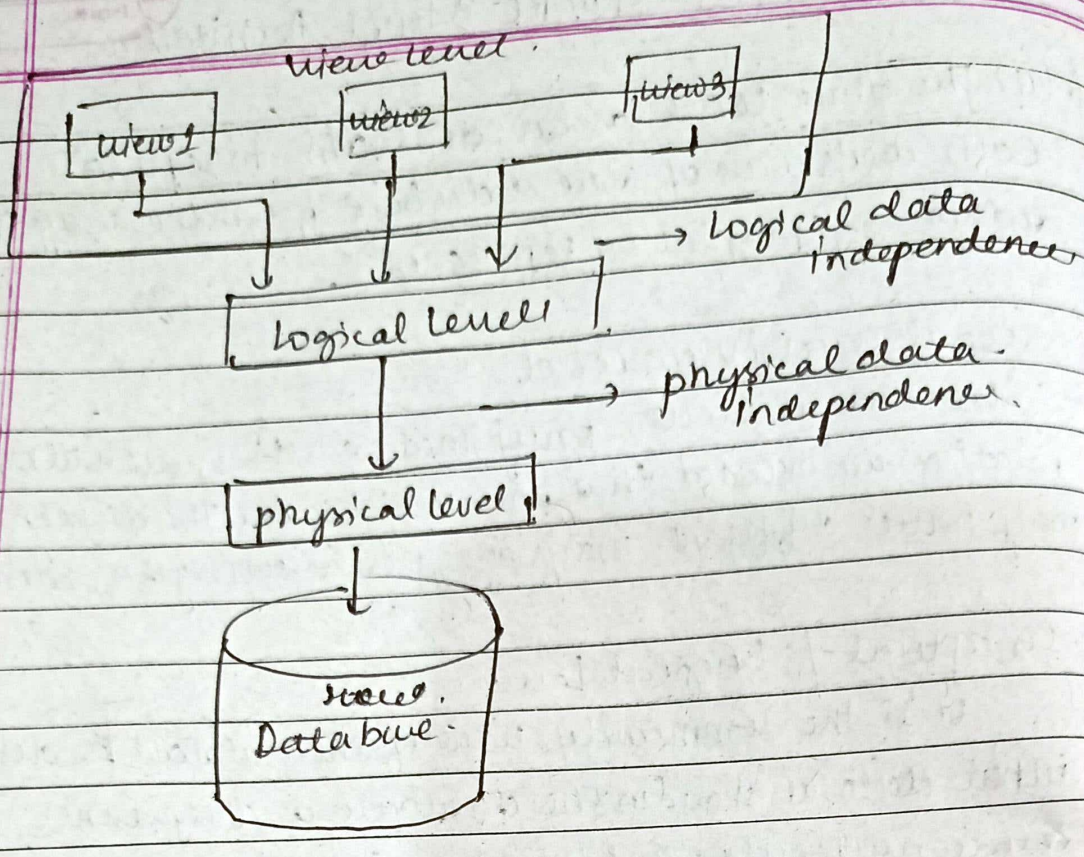
In external view only that entities, attributes, and relations are included that the user wants. The different views may have different ways of representing the same data.

(2) Conceptual / Logical level.

It is the community view of the database & describes what data is stored in the database and represents the entities, attributes and their relationships. It contains, semantic, security & integrity information about the data.

(3) Internal Lel / Physical level.

At the Physical level, the database is represented physically on the computer. It emphasizes the physical implementation of data.



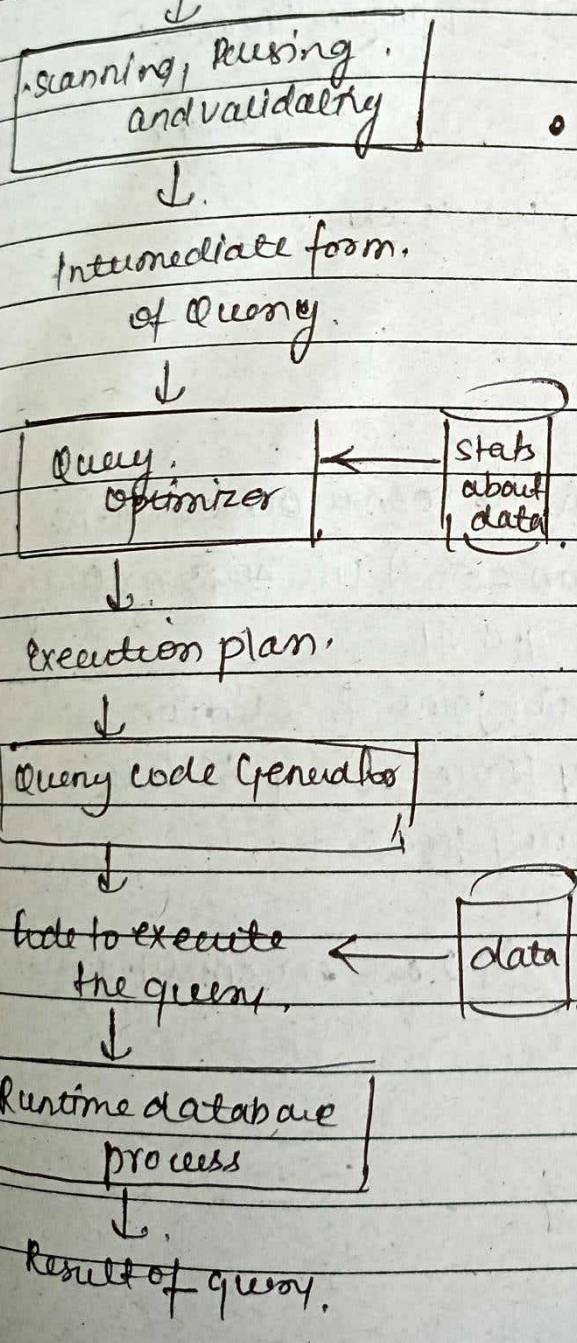
→ Logical Data Independence. Separate the conceptual level from the external level. We can change the conceptual schema without having to change the external schema.

→ Physical Data Independence. Separates the physical level: ~~data indep~~ from the conceptual level. It refers to the ability to change the physical level schema, without having to change to conceptual schema.

(b) Query Processing

Query processing is the activity performed in extracting data from one database. In query processing, it takes various steps in fetching data from database. The steps involved are:

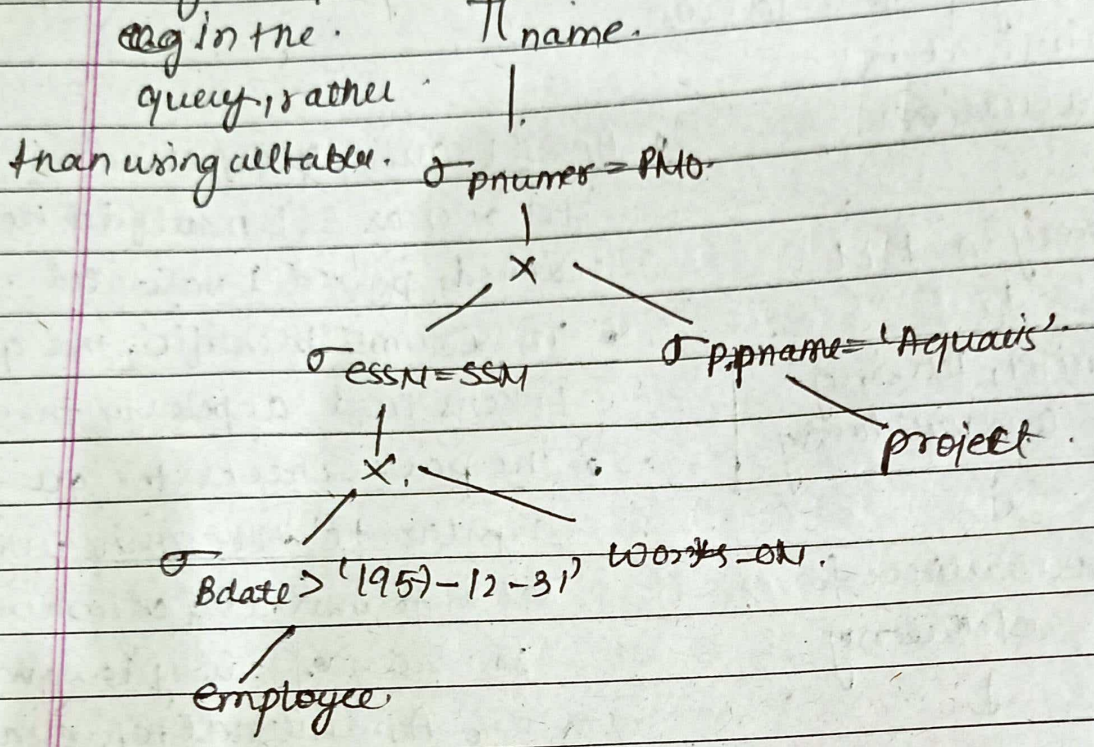
- parsing & translation
- optimization
- evaluation.



- An SQL query expressed in HLL such as SQL must first be scanned, parsed & validated.
- The scanner identifies the query tokens that appear in the query.
- The parser checks the query syntax to determine whether it is formulated according to the rules of query language.
- An internal representation of query is then created, which is executed to get the output.

(c) Heuristic Rules for Query Optimization

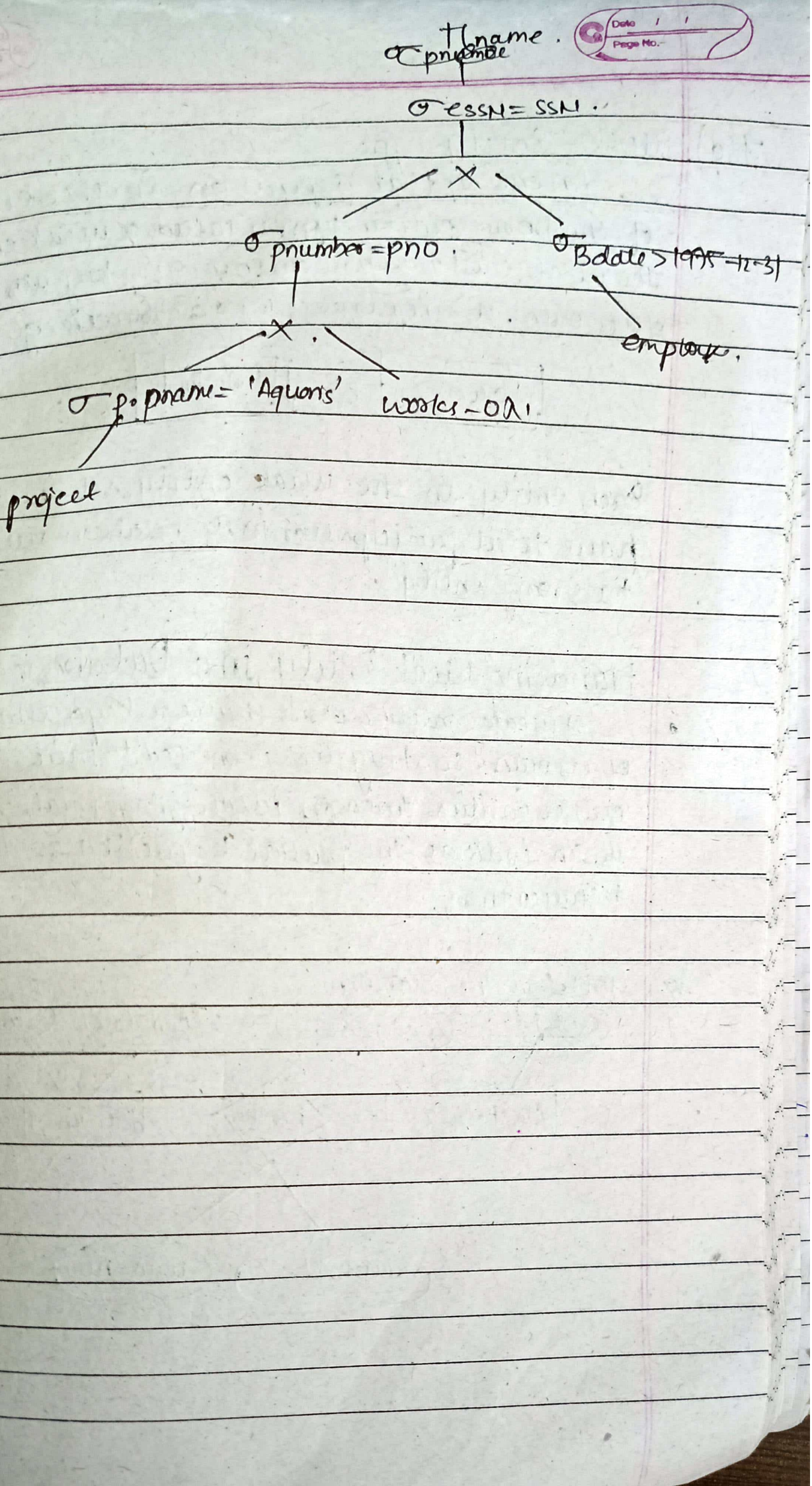
1. Perform the selection process foremost in the query. By doing so we can decrease the no. of records required in the query, rather than using all tables.



(2) Perform all projections as soon as achievable in the query. (as shown above). This helps in decreasing no. of columns in the query.

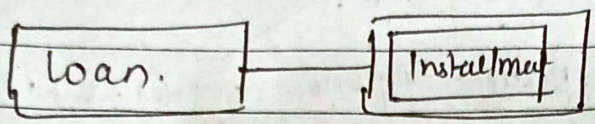
(3) Perform most restrictive joins & selection operation. Select only those sets of table that will result in less no. of records & are extremely necessary in the query.

~~The above query can be further optimized.~~


π Lname .
σ pnumber
σ essn = ssn .
×
σ pnumber = pno .
σ Bdate > 1955-12-31
×
employee .
σ p.pname = 'Aquoris'
works - on .
project

7(a). Weak - Entity Type.

An entity that depends on another entity of the strong entity type is called a weak entity. The weak entity doesn't contain any key attributes of its own. It is represented by a double rectangle.

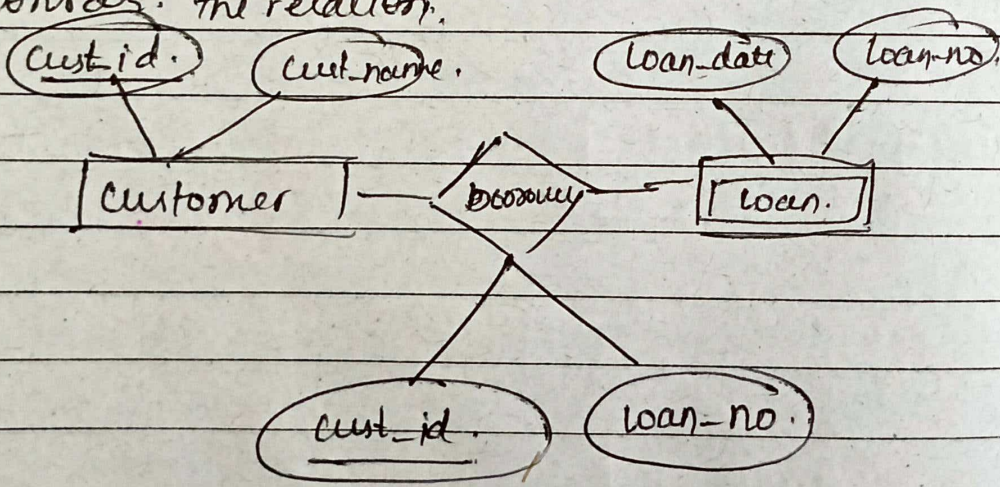


Each entity in the weak entity set must have total participation in the relation with the strong entity.

Handling Weak Entity in a Database.

- A weak entity doesn't have a key attribute of its own. So they are connected to their owner entities through identifying relationships which indicate the partial key attributes shared between them.

Consider the relation.



(b). Select operator (σ).

It is used to select the required tuples from a relation. It only selects the tuples but does not duplay them.

syntax -> $\sigma_{\text{condition}}(R)$.

e.g consider the query.

$\sigma_{C>3}(R)$.

A	B	C
1	4	4

A	B	C
1	3	3
1	4	4
3	2	3

R

(c) Project operator (Π).

It is used to project required columns from a relation. It can used along with select to put condition.

$\Pi_{(\text{attribute-list})}(R)$.

consider the query.

$\Pi_A(\sigma_{C>3}(R))$.

A
1

(c). Database Security issues.

Following.
Database security is a broad area that addresses many issues like -

(1) Various legal and ethical issue.

(2) Policy issues at governmental, institutional and corporate level.

(3). Security levels - Some organisations need to identify multiple security levels and to categorise data & users based on these classification.

Threats to databases can lead result in loss of degradation of some of all of the following commonly accepted security goals.

• Integrity • Availability • Confidentiality.

(1) Loss of Integrity →.

integrity is lost if unauthorised changes are made to the data by either Intentional or accidental act.

(2) Loss of availability →.

Data availability, refers to making the data to a human user or a program to which they have legitimate right.

(3) loss of confidentiality →

it refers to protection of data from ~~~~ unauthorised access. The loss of disclosure. The impact of unauthorised disclosure of confidential information can range from violation of data privacy act to jeopardization of national security.

Discretionary and Mandatory Access Control.

DAC → DAC is identity based access control. DAC mechanisms will be controlled by user identification such as username & password.

DAC is discretionary because the owners can transfer objects or any authenticated informations to other users. & owner can determine access privilege.

MAC → MAC provides access to user based on their identities and data. For gaining access, one user has to submit their personal information.

→ It is more secure because the rules & restrictions are imposed by admin & are strictly followed.

Discretionary Mandatory

	DAC	MAC
(1)	Easier to implement	Difficult to implement.
(2)	Less secure use.	More secure to use.
(3)	Owner can determine the access and privileges. and can restrict the resources based on identity of the users.	system determines the access ad. the resources will be restricted. based on the clearance of the. subjects.
(4)	It has higher flexibility with no rules & regulation.	Not flexible as it contains. lots of rules & regulation.
(5)	It has complete trust. In users.	MAC has trust only in administrator.

18
17 - 18
14 - 17
13 - 14
18 - 19
16 - 17

2018-19.

2(a). Data ⇒
Data refers to raw facts, figures, symbols or values that are unorganized & no meaning on their own.

(b) Record → A
A record is a collection of related data items treated as a single unit. It typically represents a single entity, containing various data fields or attributes.

(c) Information ⇒. Information is processed & organized data, that has context, meaning & relevance. It is result of interpreting & analyzing data.

(d) Database →.
A database is an organized collection of structured information, or data, typically stored in a computer system.

(e) DBMS ⇒. A DBMS is a software designed to manage, manipulate & control access to a database. It provides an interface for users & applications to interact with database.

(f) Database system →.
It refers to the combination of database along with its DBMS. It encompasses, the structures, principles & mechanisms to allow for efficient management & use of database.

(b) Database replication refers to process of creating and managing multiple copies of a database across different locations or servers in a network. These copies are synchronized to ensure consistency among them.

Database replication ~~assumes~~, ensures data availability, enhances performance, supports disaster recovery ~~ensures~~ scalability, making it a crucial aspect of modern DBMS.

(c)

~~$\sigma_{salary>5000}$ employee~~

$\Pi_{empno, fname, salary, dnum, dname}$ $(\sigma_{salary>=5000}(employee)) \bowtie_{dno=dnum} (Department)$

(d)(i) DROP ⇒ it is used to delete both the structure and record stored in the table.

eg. Drop table student;

(ii) Delete → it used to delete one or more rows from a table based on given condition.

Delete from Student where stud_id = 103;

Truncate ⇒ It deletes all records of a table, and free the space containing the table. The schema of table is retained.

e.g. Truncate table student;

(ii) where and Having clause.

Where ⇒ It is used to filter rows based on a specific condition. It is applied before any grouping of aggregations. Generally used with select, Insert, update and delete.

E.g. Select * from Student whose marks >= 60;

Having ⇒ This clause is specifically used with Group By clause. This clause filters groups of rows. It is used to apply conditions to aggregated data.

```
Select subject, count(*) as stud-count.
FROM student.
GROUP BY subject.
HAVING count(*) >= 5;
```

(e) Distributed DBMS.	Client-Server DBMS.
→ The database is shared among multiple machines. Each can act as a server & can handle requests from other systems.	Resources are stored on a server & database system can be used by the client on request.
→ more expensive.	Less expensive.
→ Resources are distributed among different systems, they can be accessed at a faster rate.	Resources are needed to be accessed by request to server. But when there are multiple requests to server, speed is reduced.
→ Database systems operates concurrently on multiple machine. Bugs in one device can be transmitted to connected devices & lead to security threat.	Proper security control mechanisms must be stored on server database to avoid security threats.

(f) Redundancy-

Data redundancy refers to repition of the similar data leading to increased storage cost, errors & compromised analytics.

Anomaly:- An anomaly is an inconsistency in the data that occurs when performing certain operations in a DBMS-

Anomalies can be caused by-

- Insertion
- Deletion
- Updation

(1) Insert Anomaly →

Insert anomaly refers to when one cannot insert a new record into a relation due to lack of data.

Consider the relation

School. ~~Student~~ (Student ID, Student Name, Student Addr, Student grades, faculty#, course).

Information of a new.

If ~~a new~~ faculty is to be added who is not assigned any course, then there could have corresponding Student_ID value which cannot be NULL.

(2) Deletion Anomaly →.

It occurs when the deletion of one record, ~~could~~ lead to the unintended loss of some other data.

eg. If a ~~to~~ faculty teaches only one student say ~~Rad~~ Ajay.

If records of Ajay are to be deleted, the faculty record will be deleted unintentionally.

(3) Updation Anomaly →.

The update anomaly occurs when updation of a ~~record~~ single data value requires updation of multiple records.

(f). The data redundancy can be minimised & anomalies be eliminated by the process of normalisation.

(g). Prime Attributes →.
A prime attribute is one which is a part of one or more candidate keys.

Non-Prime Attribute →.
Non-prime attributes are those which are not a part of any candidate key.

R(A, B, C, D, E, G, H, I).
F = {A → B, A → C, BC → D, D → EG, G → HI}.

A+ = {A, B, C, D, E, G, H, I}.

B+ = {B}
BC+ = {B, C, D, E, G, H, I}

CK = {A}.
Prime Attributes = {A}.
Non-prime Attribute = {B, C, D, E, G, H, I}

5(a). R(A, B, C, D, E, G, H, I)

P = {A → B, A → C, G → G, BC → D, D → EG, G → HI, H → AC}.

$A^+ = \{A, B, C, D, E, G, H, I\}$. ✓

$B^+ = \{B\}$.

$C^+ = \{C, G, H, I, A, B, D, E\}$. ✓

$D^+ = \{D, E, G, H, I, A, C, B\}$. ✓

$E^+ = \{E\}$.

$G^+ = \{G, H, I, A, C, B, D, E\}$. ✓

$H^+ = \{H, A, C, B, D, E, G, I\}$. ✓

$BE^+ = \{B, E\}$

B

$CK = \{A, E, D, G, H\}$.

$PA = \{A, C, D, G, H\}$.

$NPA = \{B, E, I\}$.

✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

$BCD^+ = \{B, C, D, A, E, G, H, I\}$.

$A \to B$, $A \to C$, $C \to G$, $BC \to D$, $D \to E$, $D \to G$,

$G \to H$, $G \to I$, $H \to A$, $H \to C$. Redundant.

for $A \to B$.

$A^+ = \{A, C, G, I, J, H, C\}$.

$\to A \to B$ is non-redundant.

$BC \to D$.

$BC^+ = \{B, C, G\}$.

$D \to E$, $D \to G$ are not redu.

$A \to C$.

$A^+ = \{A, B\}$.

$D \to G$

$D^+ = \{D, E\}$.

$C \to G$

$C^+ = \{C\}$.

$G \to H$

$G^+ = \{G, I\}$.

$G \to I$

$\{A \to BL,\ C \to G,\ BC \to D,\ D \to EG,$
$G \to HI,\ H \to A\}$.

$HI^+ = \{H, I, A, C, E, G\}$.

$\Rightarrow HI \to EG$ is in closure of set of FD f.

$L.H.S = EG$.

(b). (i) Entity → An entity may be an object with physical or conceptual existence.

Entity Type ⇒ It is the collection of entities that have attributes e.g student.

Entity ~~type~~ set → collection of entities of a particular entity type at a point in time.

(ii) Entity Integrity constraint uniquely identifies each record in a table. Primary key assures the entity integrity constraint is applied to a table. It cannot be a NULL value.

e.g. :

emp-id	Name	Address	Salary

The primary key emp-id uniquely identifies each row of the table.