DATE / /

DBMS VIVA

- → Tiers of Architecture
- → E-R diagram
- → Functional dependence
- → Functional Independence
- → Normalisation
- → A C I D Atomicity Isolation Durability properties
- → Transiction (Problems According to Siriazibility and & Concurring)
- → Granualarity → Shared → Enclusive → Intended
- → Locks And Unlock
  - → Shared Lock
  - → Exclusive Lock
- → Relational Algebra
- → Relational Calculas
- → Joins
  - → Natural Join
  - → Full Join
  - → Left Join
  - → Right Join
- → Time Stamping
- → Data Database backup
- → Integrity Constraints
- → Denormalization

- → Commands
- → Clauses
- → Concactination
- → Conditional Statements
- → Cursors → % ROWTYPE
- → Triggers
- → Procedure
- → Packages
- → Fundions
- →

Tiers of Architecture

Tiers of Architecture = ~~Dipep~~ On the basis of time trade off stamp, the tiers of Archicture is further divided into three parts=

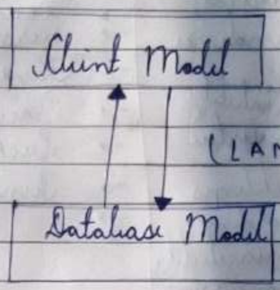
(i) Single - tier Architecture
(ii) Two - tier Architecture
(iii) Three - tier Architecture

(i) Single tier Architecture = The data is accessed directly from the database. By It's name it is clear.

B+

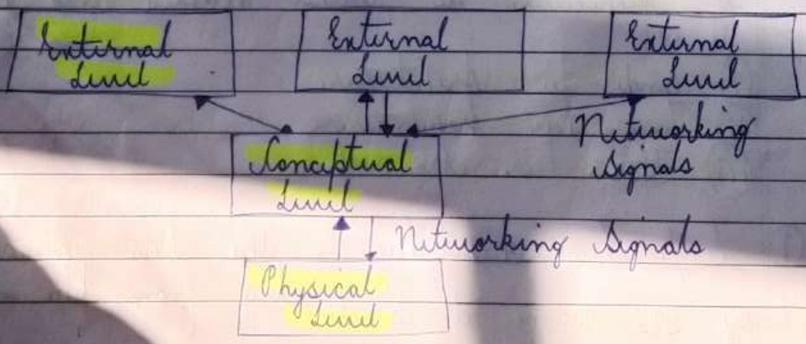
DATE ___/___/___

(ii) Two- tier Architecture = It is having combination of client model and data base model which is connected by each other through LAN.



- → Lan acts as networking signal.
- → As user/client demands the access of particular data, LAN carries it to database And retrieval of required Information from database must be done.
- → Then Again through LAN (Networking Signal) the retrieved Information is carried to client model and hence, User Accesses desired data.

(iii) Three - tier Architecture = It contains three levels :



(a) Physical Level = It deals with the physical schema like number of tables, number of attributes, number of records in a table of the database.

(b) Conceptual Level = It deals with the logical schema of the database like primary key, foreign key and ~~how~~ connection in various Attributes of the table.

(c) External Level = It deals with the view, User view. Nothing to be broke viewing about physical schema and Logical schema.

* DATA DEPENDENCE = It is a phenomena in which if data of one of the three levels is affected then all the levels are going to be affected.

* DATA INDEPENDENCE = It is a phenomena in which if data of any of the levels is affected then not going to affect the another levels.

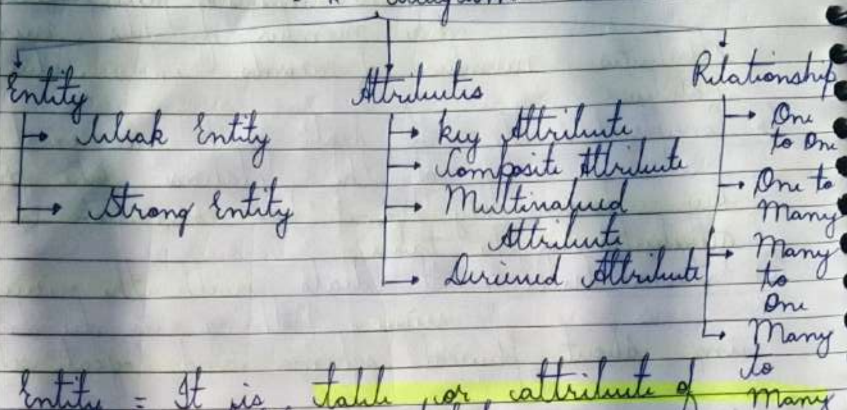
Note = Degree = Number of attributes,
Cardianality = Number of records in each Attribute.

* E - R Diagram = It stands for Entity Relationship diagram.

→ By It's name, it is very much clear that the logical and graphical representation of Entity - set is termed as E-R diagram.

* Entity Set = Group of Entity having attributes.

E-R diagram



- Entity
  - Weak Entity
  - Strong Entity
- Attributes
  - Key Attribute
  - Composite Attribute
  - Multivalued Attribute
  - Derived Attribute
- Relationship
  - One to One
  - One to Many
  - Many to One
  - Many to Many

(i) Entity = It is table or attribute of a table in database.

→ Represented by Rectangle in database.

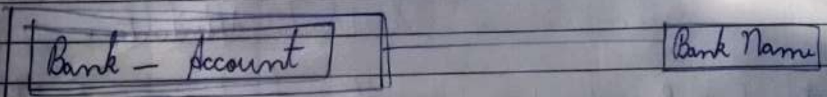
Having two types :

(a) Weak Entity = The entity which requires Another Entity In combination for Uniqueness.

→ Reliable on certain Another Entity for Uniqueness.

→ Represented by double Rectangle.

Eg = A bank Account is never represented Uniquely without bank's name.



(b) Strong Entity = The Entity which is not dependent on Another Entity for Uniqueness

→ Not reliable on Another Entity.

DATE / /

(ii) Attributes = It describes the properties of Entity (The table or attributes of a table in database).

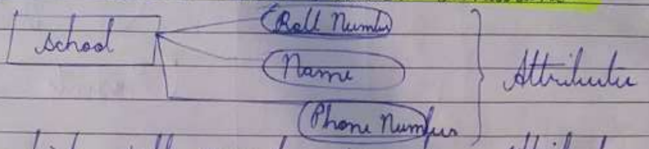
→ Represented by Oval in E-R diagram.

There are five kind of Attributes =

(i) Key Attribute = It describing the Entity set or group of Attributes uniquely

→ Represented Under Oval () with Underline.

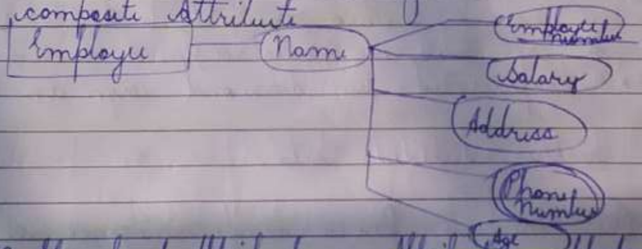
Eg =



In which roll number is key attribute.

(ii) Composite Attribute = An attribute having set of Attributes is termed as composite Attribute

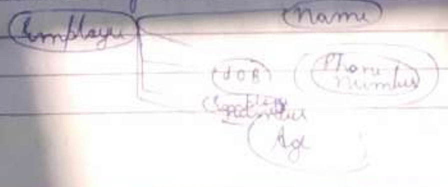
Eg =



(iii) Multivalued Attribute = Attribute that can have more than one values is termed as multivalued Attribute.

→ Represented by double Oval

Eg =

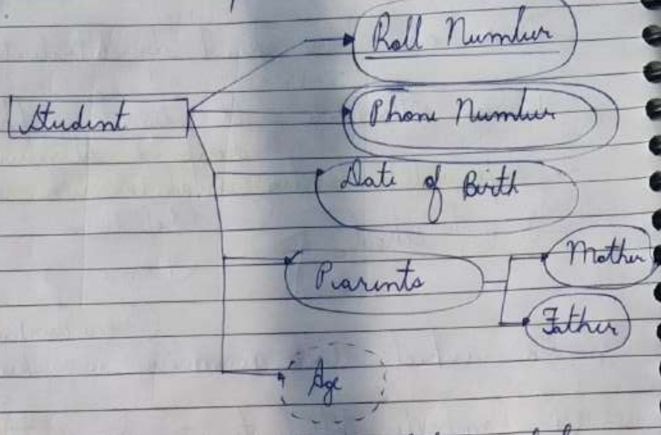


DATE / /

(iv) Derived Attribute = The attribute which is dependent upon another attribute.

→ Dashed Oval is representation in E-R diagram.

Eg =



(iii) Relationship = Describes the relation between two or Entity (A table or attribute of a table in database).

→ Represented by diamond sign in E-R diagram

Further divided Into =

(a) One to One = Single Instance of one Entity is Associated with single Instance of Another Entity.

(b) One to Many = Single Instance of one Entity is Associated with many Instance of Another Entity.

(c) Many to One = Many Instance of One Entity

DATE / /

is associated with single Instance of Another Entity

(d) Many to Many = Many Instance of One Entity is associated with many Instance of Another Entity.

* Functional Dependency = The relationship between two or mo attributes of a table is termed as functional dependency.

→ Generally occurs between primary key and a non-key Attribute.

→ The attribute which is at left side is determinant and the attribute which is at right side is functionally dependent on the determinant.

There are two types of functional dependencies =

(i) Trivial Functional Dependency = A → B, B is said to be trivial functional dependency only if B is proper subset of A.

(ii) Non-Trivial Functional Dependency = A → B, is said to be non-trivial functional dependency only if B is not proper subset of A.

→ When the Intersection of A and B comes out to be null then it becomes complete non-trivial functional dependency

DATE / /

* Keys = Useful in logical representation of Attributes.

(i) Primary key = Having Unique and Not Null property.

Eg = Employee

E-Id	E-Name	Location
1	Ram	Mumbai
2	Shayam	Mumbai
3		
4		
5		

Here, E-Id is primary key.

(ii) Candidate key = The key which has capability to become primary key, but is not a primary key. (candidacy of becoming a primary key) is termed as candidate key.

Employee

Eg =

E-Id	E-Number	Location
1	101	Mumbai
2	121	Mumbai
3	131	
4	141	
5	152	

Let's Assume E-Id as Primary key
Since, E-Number would be the Candidate key
because having tendency to become primary key.

DATE / /

(ii) Super key = Either two or more attributes in combination gets Uniquely Identified is termed as super key

(Ex) Eg =

Employee

E_Id	E_Name	Location
1	Ram	Mumbai
2	shayam	Mumbai
3	Ram	"
4	Sita	"
5	Gita	"

$\boxed{\text{E\_Id + E\_Name = Unique}}$ (Super key)

$\boxed{\text{E\_Name = Not Unique}}$

(iv) Composite key = Two or more candidate key which in combination Uniquely gets Identified is termed as composite key.

Eg =

E_Id	E_Number	Location
1	101	Mumbai
2	111	Mumbai
3	121	"
4	131	"
5	141	"

$\boxed{\text{E\_Id + E\_Number}}$ = Unique (Composite key)

(v) Alternate key = All the Attributes other than Identified primary key And composite key is termed As Alternate key.

Eg = Location In Above Example

DATE / /

Relational Algebra

- It is procedural Query Language.
- It intakes relation as Input and gives relation as Output (Works on tables)

Basic Operations

1. (i) Projection (π)
2. (ii) Selection (σ)
3. (iii) Cartesian Product ($\times$)
4. (iv) Union ($\cup$)
5. (v) Rename (ρ)
6. (vi) Set Difference ($-$)

Derived Operations

(i) Joins ($\bowtie$)

(i) **Projection** =

- Basic operation in Relational Algebra.
- Fetches whole column (Attribute)
- Represented by (π)

Syntax = $\pi_{condition}$

(ii) **Selection** =

- Basic operation in Relational Algebra.
- Fetches only particular row (record).
- Represented by (σ)
- **Syntax** = $\sigma_{(Condition)}$

(iii) **Cartesian Product** =

- We need atleast two tables for a cartesian product
- Number of rows = m x n

where, m = Number of columns rows in table1
n = Number of columns rows in table2

DATE ___ / /

Number of Columns = m + n

(iv) Union =

→ Basic Operation of Relational Algebra.

→ The uniquely Identified records in attributes of a table or set of table gets represented. (no attribute multiple times)

→ Represented by ∪ sign. ~~in~~ Relational

→ Number of attributes must be same.

(v) Rename =

→ Basic Operation of Relational Algebra.

→ Used to rename the table and is represented by ρ sign.

(vi) Set Difference =

→ Basic operation of Relational Algebra.

→ If A = {1, 2, 3, 4, 5}, B = {5, 6, 7, 8, 9}
then (A - B) = {1, 2, 3, 4}

i.e., that is, Eliminating similar record and representing the new record of only the attribute represented at left side.

→ Represented by (-) ~~symb~~ sign.

Joins =

→ It is derived Operation in Relational Algebra.

→ It is used to combine two or more common records depending on the common fields.

Of four types =

(a) INNER Join = Joins similar records or attributes of tables.

(b) LEFT Join = Joins All attributes of left table but common attribute of right table

(c) RIGHT Join = Joins common attribute left table and All Attributes of right table

(d) Full Join = Joins All Attributes of both tables and gives NULL values whose records Are not Identified.

* INTEGRITY Constraints

Entity Integrity Constraint =

→ The tuples should be Uniquely Identified by Primary key.

Referential Integrity Constraint =

Foreign key

Check Constraints =

→ Whether the value is within given constrain range or not.

Eg = Age

Default Constraints =

→ If no value is Entered, then default value is represented.

Eg = Semester of particular Amount of students in University. ([illegible])

DATE / /

UNIT - 3

(i) Transiction Properties

~~ACF~~ Atomicity
- → Consider It as a single cell
- Eg = If ₹1000 is withdrawn from A's Account, then ₹1000 will be deposited to B's Account.

Constitincy
- → The data at particular time period should be Accurate
- → The total Amount before transaction will be equal to total Amount After transaction

Isolation
- → Only the performance of read and read function
- No other operation can be performed.

Durability
- → Partial transaction should be Aborted.
- → The commit will only Occur when transaction becomes complete.

Transiction Execution =

(i) Serializibility = Transictions Occuring in sequenced manner.

(ii) Concurrency = Transactions Occuring parallely due to which there are certain problems.

- Temporary Update Problem
- Incorrect Summary Problem
- Lost Update Problem
- Unrepeatable Problem

DATE / /

(i) Temporary Update Problem =

- When two transactions are running concurrently (parallely).
- Then If one got Aborted, it will get reverted And will nullify it's Effect but the second transaction has Already read the data
- Hence,
- there Where Inconstitency Occurs

(ii) Incorrect Summary Problem =

- When two transactions are running parallely in combination of Aggregate functions then when one of them gets aborted and gets revert back by nullifying its Effect
- The second transaction Already read, ~~the~~ hence, will become Inconstitent

(iii) Lost Update Problem =

- The Update by One transaction got lost due to overwritten as Update is Also done by second transaction

(iv) Unrepetable Problem =

- The data read by both the transactions at same time is same but their values are not same, hence the occurence of Inconstitency will be there

Time Stamping =

- Every time stamp must have Unique Identification.

DATE / /

- The priority is decided as smallest timestamp will get highest priority
- The timestamp will get comparable on the basis of sharable mode
  1. Read Mode ⟶ Sharable Mode
  2. Write Mode ⟶ Exclusive Mode
  3. Execute Mode ⟶ Dependent
- T1 will run first and then T2 and T2 can also run first and then T1.

Locks :

There Are two type of locks

(i) Shared Lock : If any one transaction is acquiring the lock and any other transaction needed it, then sharing must happen between locks

Growing Mode :

- Locking all the operations to be performed in transaction

Shrinking Mode :

- Unlocking the locked operations performed on transactions (data items)

Note : If locking then full lock and then Unlocking.

- If Unlocking then full Unlocking and then lock.

(ii) Exclusive Lock : If any transaction is Acquiring the lock and any other transaction also requires that then not possible.

DATE / /

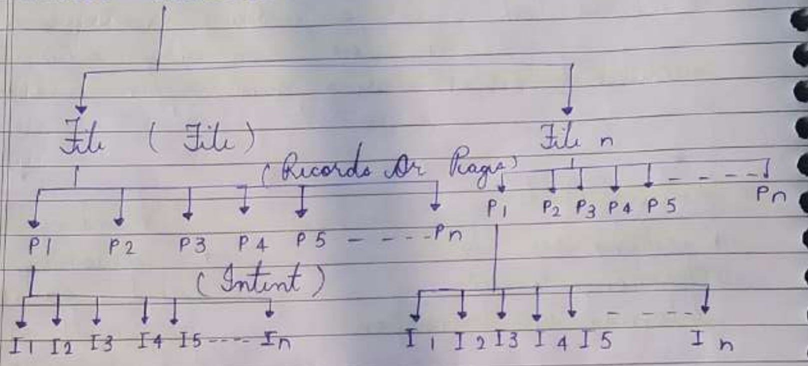
Multiple Granularity System =

Fine Granularity =

→ Within database is divided into small pieces or granule, then change in one part of granule will not affect the whole database.

Coarse Granularity =

→ Within database is divided into pieces of granule then changes in one part of granule will Affect the whole database



Shared ⇒ Top Node of Hirercahy
Exclusive
Intent ⇒ Trying to Access least level Nodes or discendent Nodes.

Database Recovery = Due to any concurrent Transaction or recovery, data becomes Inconsistent. Hence, to rectify and recover these

DATE / /

various different database recovery methods are used =

(i) Log Based Recovery =

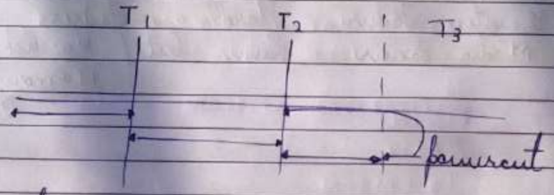
→ Recovery based upon date and time, updation value, Type of operation, file, record or Intent

(ii) Database Recovery =

→ Due to Any hardisk, software, or powercut, backup And recovery is required.

(a) Differed Backup =

→ Unless and Until, transaction becomes complete, the updation does not takes place.



Advantages =

→ Reduces lifecycle of processor.

Disadvantages =

→ The whole transaction has to start again.

(b) Immediate Backup =

→ After each and every operation, backup has been taken.

Disadvantage =

→ Not directly connected with central processing unit, hence timetaking

DATE / /

(iii) Shadow Recovery =

→ Unless and until there is complete transaction, the page is not going to get reflected

* BACKUP =

Full =

Not preferrable

Partial =

→ Only Important piece of operation is kept at backup.

Since, memory management is convenient.

Types of Failures =

(i) Transiction Failure = Software Failure

(ii) System Failure = Hardware Failure

(iii) Media Failure = Power Cut / Hardisk Failure / Memory Failure

DATE / /

UNIT - 2

* Normalization = Decomposition of Big Tables into smaller table given

Conditions =

(i) Should not have Any Anamoly like Insertion Anamoly, deletion Anamoly, Updation Anamoly so that data Inconsistency does not occurs.

1NF = Every cell contain Only one Atomic value.

2NF = 1NF + Every non-key attribute should be irreducably dependent upon primary key

3NF = 2NF + Every non-key attribute should be non-transitively dependent on primary key.

BCNF = 3NF + determinant is super key

4NF = BCNF + no multivalued attributes (dependencies)

5NF = 4NF + no join dependencies having joining as lossless.

→ Used to reduce the redundancy and duplicacy

* Denormalization = In denormalization, redundancy and duplicacy is added due to quick execution of the query.

DATE / /

*. **Triggers** = The triggers Automatically gets Invoked of before or After certain DML Events such As Insertion, deletion, updation.

Syntax =

```
create trigger trigger_name before/
                                      After
  INSERT / update/delete
   on
   table_name
    For each row
    // Trigger body.
```

*. **Procedures** = Used to perform specific task

(i) IN = Used to send values to Identified process.

(ii) OUT: Used to get values from Identified process.

* **Syntax** -

```
create or replace procedure procedure
name ({IN, OUT, INOUT} ~~Decler~~ datatype)
IS
DECLERATION SECTION
BEGIN
//BODY
EXCEPTION
END;
```

*. **Packages** = It is group of functions or procedures

Syntax = Package Declaration
(Prototype of function or Package)

Package definition

Package Call

DATE / /

Cursors =

→ In SQL cursors hold multiple rows returned by SQL Statement

→ There are two types of cursors

(i) Implicit Cursor = Generated by Oracle

(ii) Explicit Cursor

Declare → CURSOR C1 IS Select statement
Open → OPEN C1
Fetch → FETCH C1
Close → CLOSE

% TYPE = defines type of compatibility between type ~~of data~~ columns in table.

Before first fetch from an open cursor
cursor_name % TYPE returns returns null and after first fetch If returns row then true and if returns ~~column~~ not row. Then false.

% ISOPEN = If Cursor is open the cursor_name % ISOPEN returns true.

If cursor is closed then cursor_name % ISOPEN returns false.

% NOTFOUND = Before first fetch returns NULL.
If After first fetch returns row successfully then returns false value.
If After first fetch the row is not fetched successfully or gets failed then returns false value.

% ROWTYPE