Unit - 1

E.R. Model

E.R. Model is a widely used High-level data model. It is used for conceptual designing of database applications and other database tools. The ER Model is based on Real World entities and their associations / Relations with each other. The E.R. Model uses well-defined symbols to design ER Models.

Symbols used in ER Model

i)  → Entity

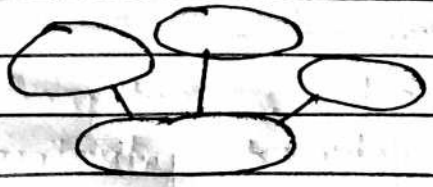
ii)  → Weak Entity

iii)  → Relationship

iv)  → Identifying Relationship

v)  ⇒ Attributes

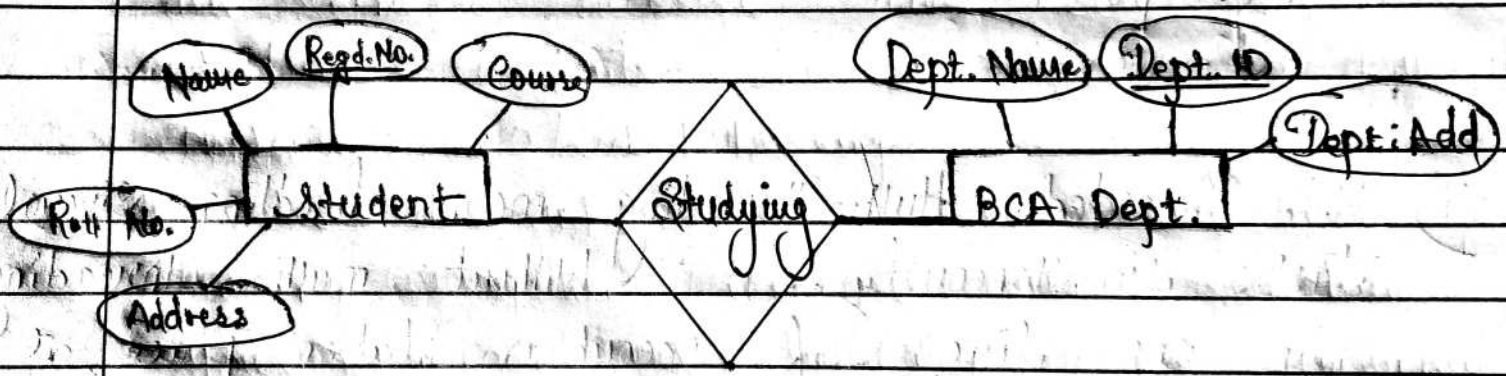
vi)  ⇒ Key Attributes

vii)  ⇒ Composite Attributes

viii)  ⇒ Multivalued Attribute

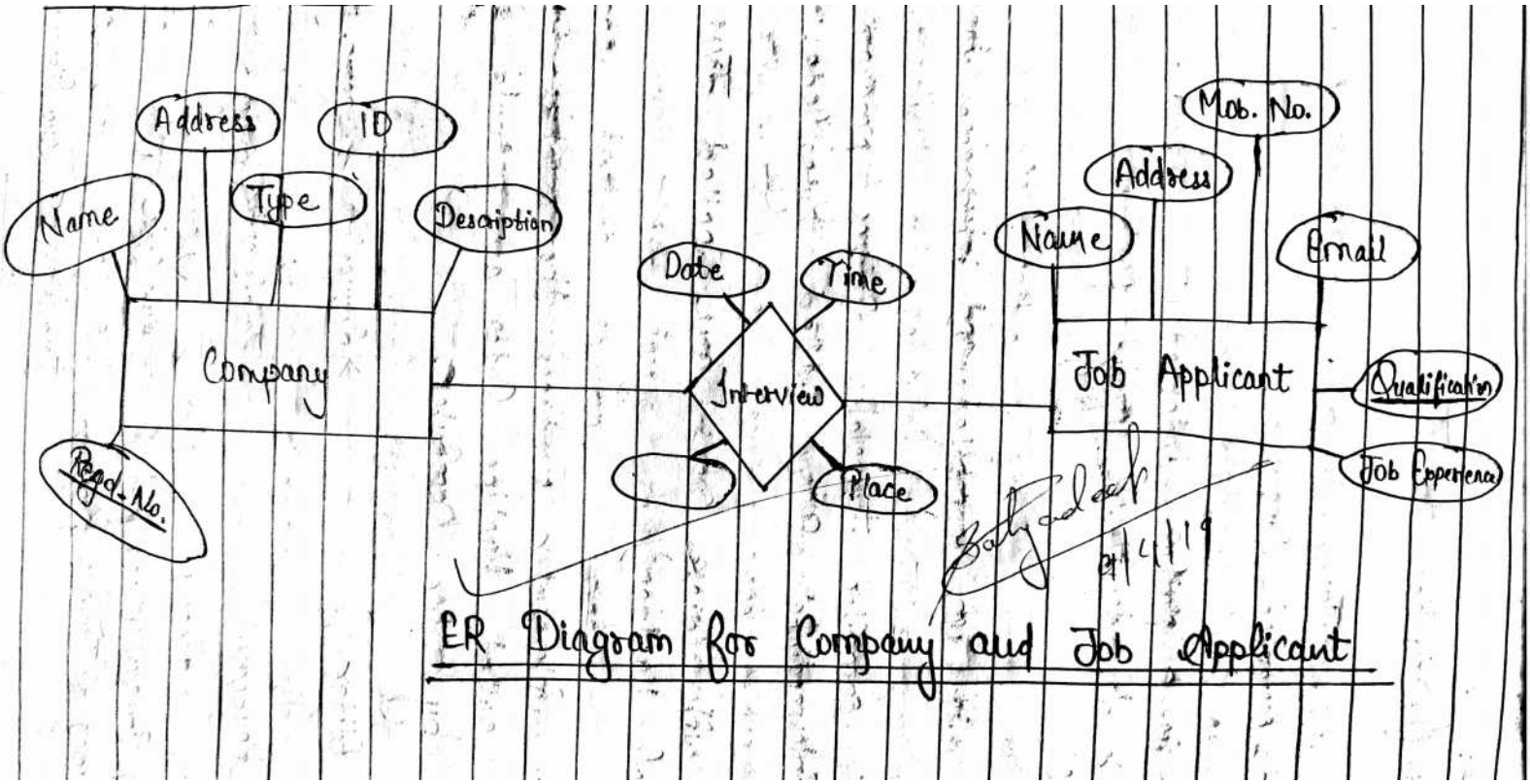
ix)  ⇒ Derived Attribute

Code: 5BCA4
Subject: ORACLE DBMS
Topic: ER Diagram for Student and Department



H.W.

Draw an ER Diagram for Company and Job applicant.



ER Diagram for Company and Job Applicant

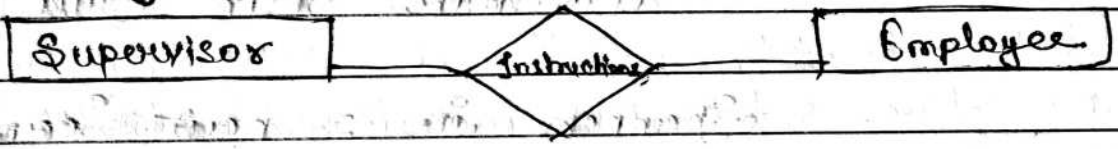
Subject : ORACLE RDBMS
Topic : Relation between Entities

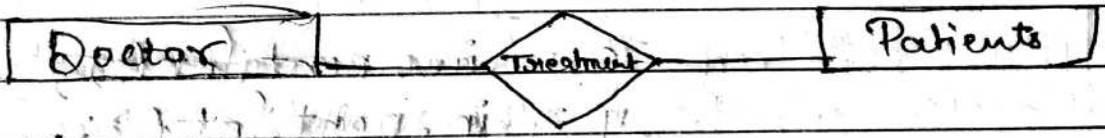
Binary Relationships

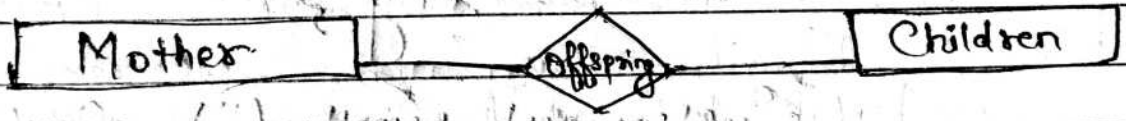
Binary Relationships exist b/w 2 entities. It is divided into 3 types :-

One-to-One :-

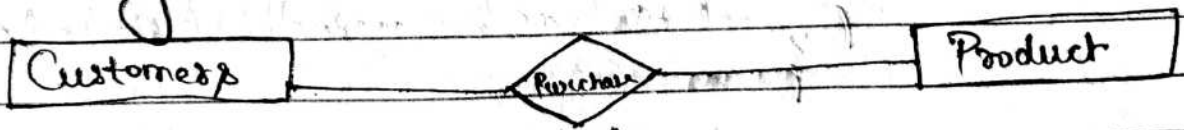
One-to-Many :-

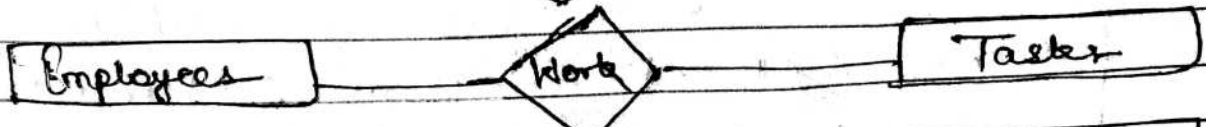


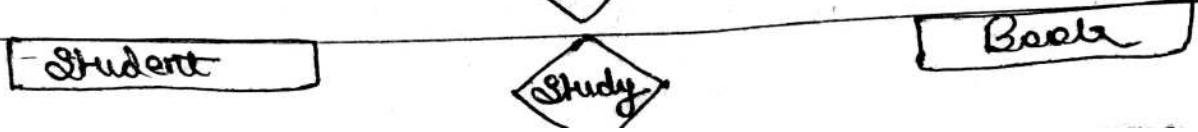




Many-to-Many :-







Data Abstraction, Generalization and Specialization:-

Data Abstraction:-

Database Systems / Relational Database Systems are made up of complex structures which if introduced to the user, then it may turns out to be ardous task for them. To ease the user interaction with database, the developer hides internal irrelevant structural details from the user. The process of hiding this irrelevant structure details from the user is known as "Data Abstraction".

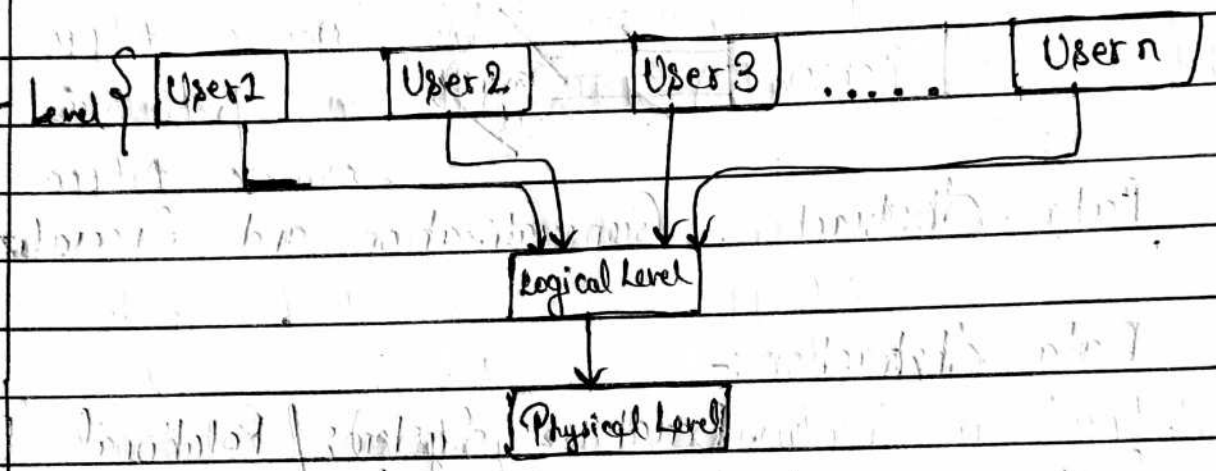
There are three levels of 'data abstraction':-

i) Physical level:- This is the lowest level of

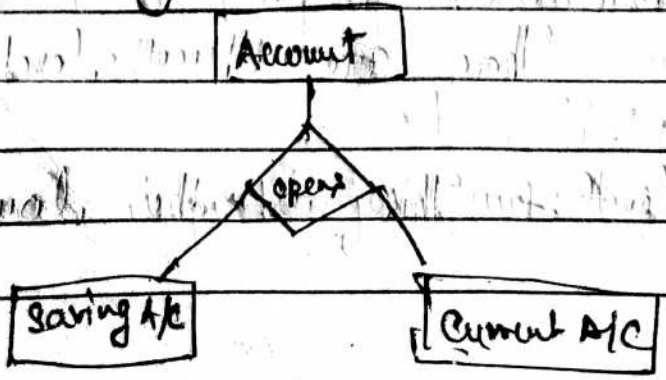
of data abstraction. It is used to describe how data is actually stored in the database.

ii) Logical Level :- This is the middle level of data abstraction. It is used to describe detail data structures ~~the~~ used in the database.

iii) View Level :- This is the highest level of data abstraction. This level describes how the user interacts with database.

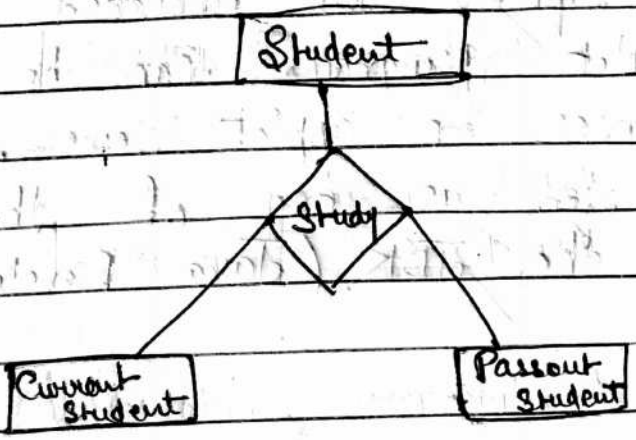


Generalization :- The term Generalization is used to refer to the process of defining a generalized entity type from the given entity types. It is a bottom-up approach in which the lower level entities combine to form a higher entity.



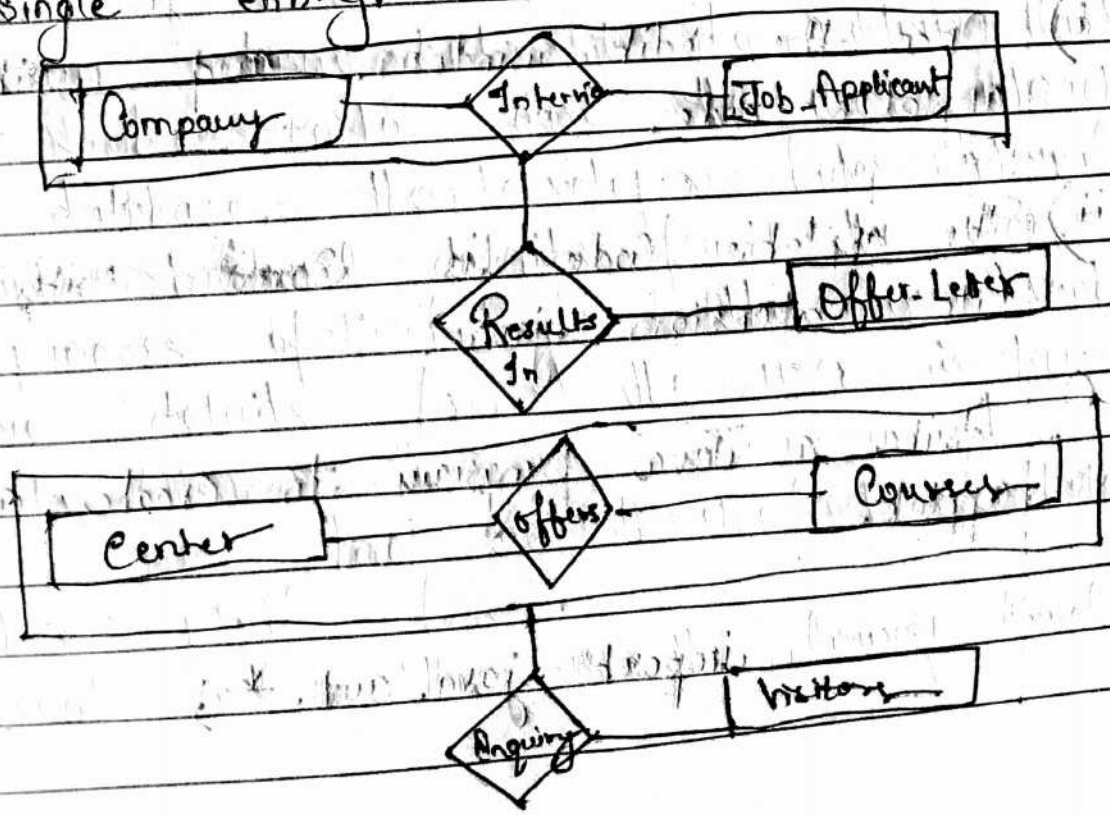
Specialization :-

Specialization is reverse of generalization. It follows top-down approach in which one higher level entity is broken down into two or more lower level entities. Some higher level entities may or may not have lower level entities or set at all.



Aggregation :-

Aggregation is a process in which relationship b/w 2 entities is treated as single entity.



Relational Algebra :-

Relational Algebra provides the basic set of operations for relational model. It provides formal foundation for relational model operations. It is used for implementing and optimizing queries in RDBMS.

following are the relational algebra operations :-

i) SELECT Operation: SELECT Operation (σ) is used to Select the tuples from the relation that satisfies the given condition :-

Example :-

$\sigma_{salary > 20,000}$ (Employee)

$\sigma_{<Condition>}$ (R)

ii) Projection Operation (π)

The projection operation is used to select the required columns from the given table and discard other columns. If we require only certain attributes of a relation, then we use projection operation.

Example :-

$\Pi_{(FName, LName, Salary)}$ (Employee)

$\Pi_{<List\ of\ Attributes>}$ (R) General form of Projection Operation

Read as (R here)

iii) Rename Operation (ρ)

It is used to rename the relation name or attribute name.

$\rho_{S}(A_1, A_2, A_3, \ldots)$ (R)

where S is the new relation name, and $A_1, A_2, \ldots$ are new attribute name.

iv) Union Operation (∪):
It is used to merge all the attributes of relation R_1 and R_2.

$R_1 \cup R_2$

v) Intersection Operation (∩):
It is used to select those attributes which are common to both relation R_1 and R_2 and form a new relation.

$R_1 \cap R_2$

vi) Difference or Minus Operation (−):
In this operation, in which all tuples (records) which are present in Relation R but not in relation S which is taken into consideration.

$R_1 - S$

vii) Cartesian Product / Cross Product Operation (X)
A relation $R_1 \times R_2$ produces such a relation that has all the attributes of R_1 and R_2 and includes all possible combinations of all possible combinations of R_1 and R_2.

R_1 (1, 3, 5)
R_2 (2, 4, 6)

$R_1 \times R_2$ (12, 14, 16,
32, 34, 36,
52, 54, 56)

Data Independence:

Data Independence is the ability to make changes in data characteristics without having to make changes to the programs that access the data. It is very important feature of data because it saves enormous time and errors caused by modification which takes place through the software.

Referential Integrity and Database Integrity:

Referential Integrity (R.I.) is a relational database concept which states that table relationship must always be consistent. R.I refers to the accuracy and consistency of data within a relationship. In a relationship, a data is linked b/w two or more table. This is achieved by having different types of keys in other words, any foreign key field must agree with the primary key field of the first table.

Database Integrity is a concept which states that all the databases and relationship must always be consistent. This is achieved and implemented by using and giving special rights and previliges to the Database Administrator.

SQL (Structured Query Language)

SQL is used to access database records & information. It is independent of any database application s/w. All the database support SQL with little with modification in it syntax.

* The SQL has three parts :-

(1) DDL (Data Definition Language) → It is use to define the database structure.

(2) DML (Data manipulation language) → It is use to modify & alter the given data.

(3) DCL (Data control Language) → It is use to control the database. DCL commands is are use by DBA (Database administrator).

SQL (Structured query language, Sequel)

IBM develop the original version of SQL. As part of the system R project in the early 1970's. Many of the products now support SQL language. SQL has established itself as the standard relational database language.

(1) DDL (Data Definition Language) → DDL provides commands for defining relational database

Schemas, deleting relations & modifying relations.

(i) Create commands:-

* Syntax:

Create table <table name> (A1, A2, A3... An, Integrity constraints (A i));

* where, A1, A2 ... are attributes of the table.

* example →

create table customer
customer_name varchar (20),
customer_street char (30),
customer_city char (30),
Primary key (customer_name),

customer name	customer stree	customer city

→ Schema

Create commands create a Database schema. It means to create the structure of the table.

(ii) Drop command:-

* Syntax *

Drop table <table_name>

* e.g., Drop table customer.

* Drop command is use to remove the relation from database.

(iii) Alter command :-

* Syntax :-

Alter table <table_name> add A
Or Alter table <table_name> Drop D

* e.g.,

Alter table customer add customer i'd.
Alter table customer drop customer i'd.

(2) DML (Data manipulation language) →

(a) Select command :- Syntax :-

select <attributes>
from <relation / Table_name>
where condition

* e.g., select customer_name, customer_street
from customer
where customer_city = patna

(b) Insert command :-

* Syntax :-

Insert into <Table_name>

* e.g., Insert into customer values ("Amit", "Boring road", "Patna").

(C) Delete command :-

* Syntax: Delete from <Table name>
where <condition>

* e.g., Delete from customer
where customer-name = Amit.

(d) Update command :-

In certain situation, we want to change a value in a tuple without changing all the values in the tuple. For this purpose, we use update command.

* Syntax :- Update <table-name>
where (condition)

* e.g., update a/c
set balance = balance * 1.05
where balance > 1000

(3) DCL (Data control language) →↓←

DCL commands are use by database administrators to keep the database safe & secure any unauthorised user. Grant & Revoke are common command use by DBA.

UNIT - 2

Normalisation is a process of decomposing tables to eliminate data redundancy (Repeatation) & undesirable characteristic like Insertion, anomaly, Updation anamoly & Deletion anamoly. Normalisation is a multistep process that puts table into tabular form by removing duplicate data from the table.

* There are four normal forms :

(a) First normal form (1NF).
(b) Second normal form (2NF).
(c) Third normal form (3NF).
(d) fourth Normal form (BCNF).

(a) First normal form (1NF)

Rule: "Each attribute in the table must have atomic (single) values."

emp.id	emp-name	emp add	emp-mob
101	Rohit	Dumri	012345
102	Kunwar	Dumri	0162850

Employee (Table 1NF)

(b) Second Normal form (2NF)

Rule: A table is said to be in 2NF. If it holds -

(i) The table is in 1NF. No non prime attribute is

* Note:
Non-prime attributes :- An attribute that is not part of any candidate key is known as non-prime attribute.

* e.g.,

Teacher id	subject	Age
101	Math	16
102	Physics	35
103	Chemistry	60

Candidate key : Teacher_id, subject
non-prime attributes - Age.

* This table is in 1NF because each attribute has atomic values. However, it is not in 2NF because non-prime attribute teacher "age" is dependent on Teacher-Id. These violates the rule that says "No non-prime rule that says". No non prime attribute is dependent on the proper subset of any candidate key.

* To make the table follow 2NF we break it into two tables.

Table: 2NF

Teacher-id	Teacher age
112	54
114	48

Teacher-id	sub
112	math
114	eng.

(c) 3NF :-

Rule: (i) Table must be in 2NF.
(ii) Transitive functional dependency of non-prime attributes on any super key should be removed.

* Super key *

Super key is a set of one or more attributes to uniquely identify a tuple in a relation.

Transitive functional Dependency

Transitive functional dependency can only occurs in a relation of three or more attributes. These dependency helps us to normalise the database in the 3NF. A functional dependency is set to be transitive if it is formed by two functional dependencies.

* e.g., $x \to z$ is a transitive FD. If the following three FD holds true —

$x \to y$

y does not $\to x$

$y \to z$

Book (x	Author (y)	Author Age
Let us C	Yashwant Kanitkar	35
N/w	Frozen	60
Database	Korth	62

(i) $x \rightarrow y$

Book → Author

(If we know the Book, we know the author

(ii) y does not $\rightarrow x$

(If we know the author, we doesn't know the book.

{Author} does not {BOOK}.

(iii) $y \rightarrow z$

{Author} → {Author Age}

* Let us consider student detail table.

(i)

Stu-i'd	Stu-Name	Street	City	State	Pin

* In these table, Student-i'd is primary key but street, city & state depend on PIN number. The dependency b/w PIN number & other fields is called as Transitive dependency. Hence, for 3NF we need to move the street, city & state to new table with PIN number as primary key.

(ii)

Stu-i'd	Stu-Name	PIN
101	Rohit	802120
102	Aman	802120

"New Student Detail"

(iii)

PIN	STREET	CITY	STATE
802120	Dumri	Buxar	Bihar
802120	Dumri	Buxar	Bihar
802120	Dumri	Buxar	Bihar

→ Address detail ←

* Advantage of Transitive functional dependency

(i) Amount of data Publically is removed
(ii) Data integrity is achieved.

(iv) BCNF (Boyce codd normal form) →

It is also known as 3.5NF. It is slightly stronger version of 3NF. BCNF was developed in 1974 by Raymond F. Boyce and Edgar F. Codd. If a relational schema is in BCNF than all the redundancy base on functional dependency is remove. A relational schema are is in BCNF if & only if $X \to Y$ & X is a superkey.

Join Dependency

If a table can be recreated by joining multiple tables and each of this table have a subset of the attributes of the table, then the table is in Join Dependency. It is a generalization of Multivalued Dependency

Join Dependency can be related to 5NF, wherein a relation is in 5NF, only if it is already in 4NF and it cannot be decomposed further.

Example

<Employee>

EmpName	**EmpSkills**	**EmpJob** (Assigned Work)
Tom	Networking	EJ001
Harry	Web Development	EJ002
Katie	Programming	EJ002

The above table can be decomposed into the following three tables; therefore it is not in 5NF:

<EmployeeSkills>

EmpName	**EmpSkills**
Tom	Networking
Harry	Web Development
Katie	Programming

<EmployeeJob>

EmpName	**EmpJob**
Tom	EJ001
Harry	EJ002
Katie	EJ002

<JobSkills>

EmpSkills	**EmpJob**
Networking	EJ001
Web Development	EJ002
Programming	EJ002

Our Join Dependency:

{(EmpName, EmpSkills), (EmpName, EmpJob), (EmpSkills, EmpJob)}

The above relations have join dependency, so they are not in 5NF. That would mean that a join relation of the above three relations is equal to our original relation **<Employee>**.

Multivalued Dependency

- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.

Example: Suppose there is a bike manufacturer company which produces two colors(white and black) of each model every year.

BIKE_MODEL	MANUF_YEAR	COLOR
M2011	2008	White
M2001	2008	Black
M3001	2013	White
M3001	2013	Black
M4006	2017	White
M4006	2017	Black

Here columns COLOR and MANUF_YEAR are dependent on BIKE_MODEL and independent of each other.

In this case, these two columns can be called as multivalued dependent on BIKE_MODEL. The representation of these dependencies is shown below:

1. BIKE_MODEL → → MANUF_YEAR
2. BIKE_MODEL → → COLOR

This can be read as "BIKE_MODEL multidetermined MANUF_YEAR" and "BIKE_MODEL multidetermined COLOR".

Query Processing

Query Processing would mean the entire process or activity which involves query translation into low level instructions, query optimization to save resources, cost estimation or evaluation of query, and extraction of data from the database.

Goal: To find an efficient Query Execution Plan for a given SQL query which would minimize the cost considerably, especially time.

Cost Factors: Disk accesses [which typically consumes time], read/write operations [which typically needs resources such as memory/RAM].

The major steps involved in query processing are depicted in the figure below;

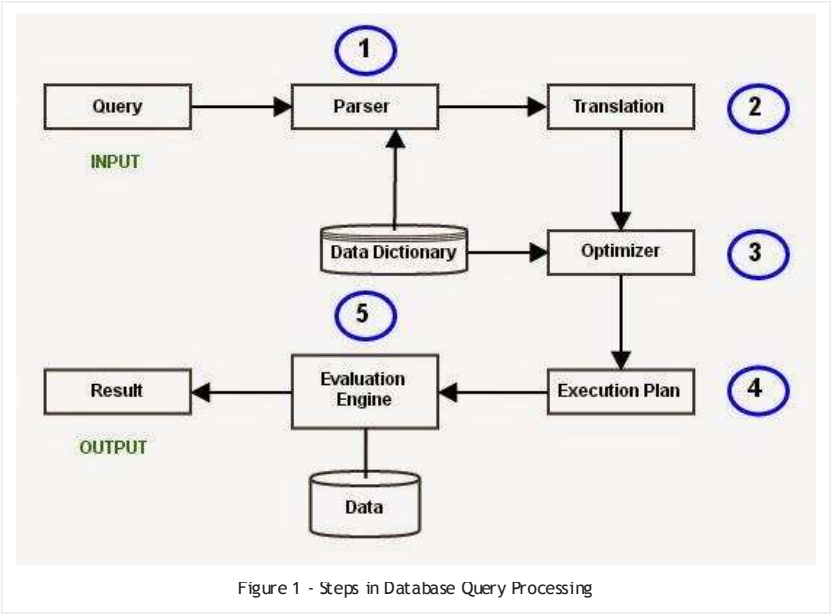


Figure 1 - Steps in Database Query Processing

Let us discuss the whole process with an example. Let us consider the following two relations as the example tables for our discussion;

Employee(Eno, Ename, Phone)

Proj_Assigned(Eno, Proj_No, Role, DOP)

where,

Eno is Employee number,

Ename is Employee name,

Proj_No is Project Number in which an employee is assigned,

Role is the role of an employee in a project,

DOP is duration of the project in months.

With this information, let us write a query to find the list of all employees who are working in a project which is more than 10 months old.

```
SELECT Ename
FROM Employee, Proj_Assigned
WHERE Employee.Eno = Proj_Assigned.Eno AND DOP > 10;
```

Input:

A query written in SQL is given as input to the query processor. For our case, let us consider the SQL query written above.

Step 1: Parsing

In this step, the parser of the query processor module checks the syntax of the query, the user's privileges to execute the query, the table names and attribute names, etc. The correct table names, attribute names and the privilege of the users can be taken from the system catalog (data dictionary).

Step 2: Translation

If we have written a valid query, then it is converted from high level language SQL to low level instruction in Relational Algebra.

For example, our SQL query can be converted into a Relational Algebra equivalent as follows;

$$\pi_{Ename}(\sigma_{DOP>10 \wedge Employee.Eno=Proj\_Assigned.Eno}(Employee \times Prof\_Assigned))$$

Step 3: Optimizer

Optimizer uses the statistical data stored as part of data dictionary. The statistical data are information about the size of the table, the length of records, the indexes created on the table, etc. Optimizer also checks for the conditions and conditional attributes which are parts of the query.

Step 4: Execution Plan

A query can be expressed in many ways. The query processor module, at this stage, using the information collected in step 3 to find different relational algebra expressions that are equivalent and return the result of the one which we have written already.

For our example, the query written in Relational algebra can also be written as the one given below;

$$\pi_{Ename}(Employee \bowtie_{Eno} (\sigma_{DOP>10}(Prof\_Assigned)))$$

So far, we have got two execution plans. Only condition is that both plans should give the same result.

Step 5: Evaluation

Though we got many execution plans constructed through statistical data, though they return same result (obvious), they differ in terms of Time consumption to execute the query, or the Space required executing the query. Hence, it is mandatory choose one plan which obviously consumes less cost.

At this stage, we choose one execution plan of the several we have developed. This Execution plan accesses data from the database to give the final result.

In our example, the second plan may be good. In the first plan, we join two relations (costly operation) then apply the condition (conditions are considered as filters) on the joined relation. This consumes more time as well as space.

In the second plan, we filter one of the tables (Proj_Assigned) and the result is joined with the Employee table. This join may need to compare less number of records. Hence, the second plan is the best (with the information known, not always).

Output:

The final result is shown to the user.

The overall information discussed above are depicted in Figure 2:

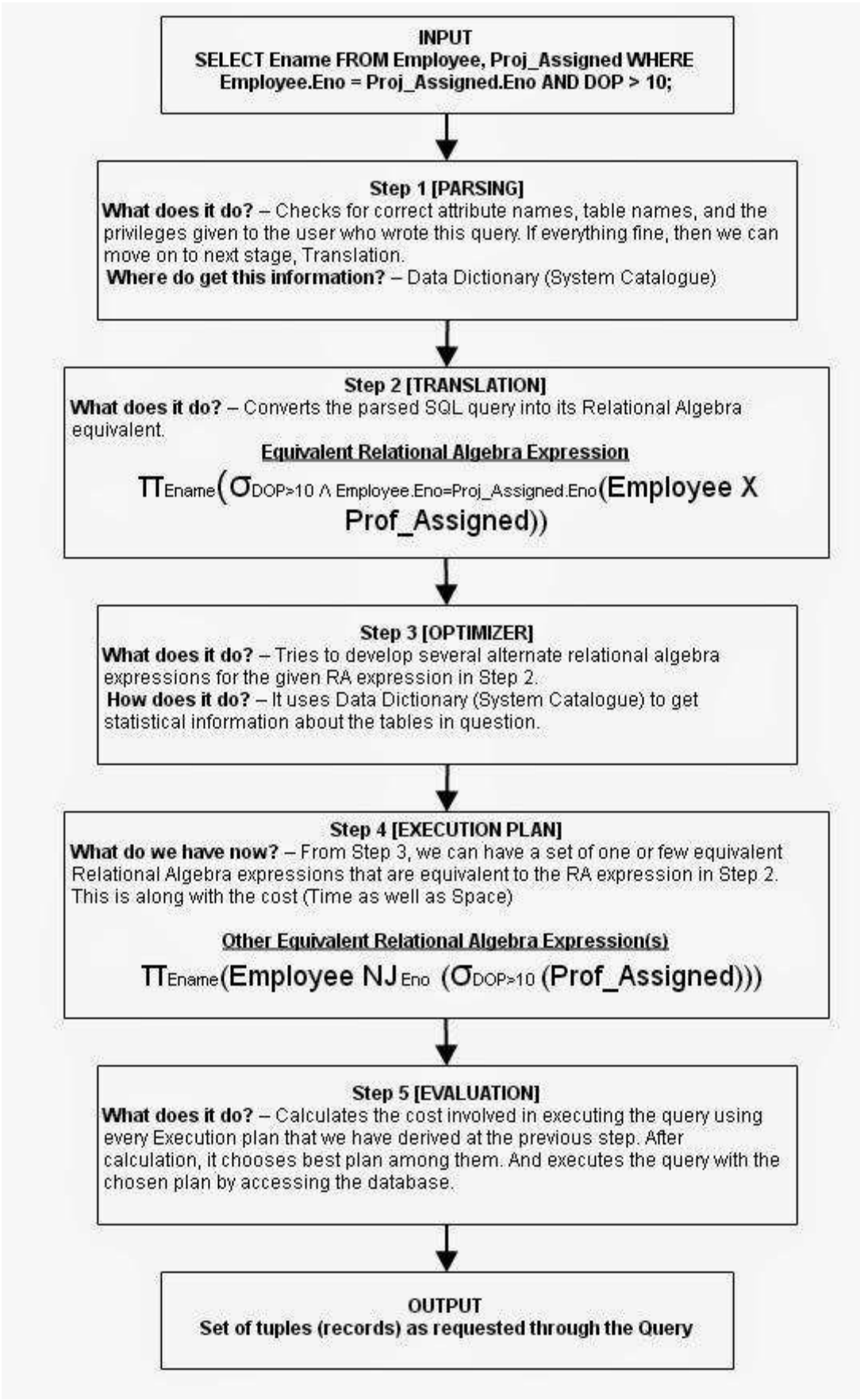


Figure 2 - Query Processing [Note: in Step 4, NJ means Natural Join]

UNIT-3

Concurrency:

Concurrency is the ability of a database to allow multiple users to affect multiple transactions. This is one of the main properties that separates a database from other forms of data storage like spreadsheets.

The ability to offer concurrency is unique to databases. Spreadsheets or other flat file means of storage are often compared to databases, but they differ in this one important regard. Spreadsheets cannot offer several users the ability to view and work on the different data in the same file, because once the first user opens the file it is locked to other users. Other users can read the file, but may not edit data.

Concurrency Control

- In the concurrency control, the multiple transactions can be executed simultaneously.
- It may affect the transaction result. It is highly important to maintain the order of execution of those transactions.

Problems of concurrency control

Several problems can occur when concurrent transactions are executed in an uncontrolled manner. Following are the three problems in concurrency control.

1. Lost updates
2. Dirty read
3. Unrepeatable read

1. Lost update problem

- When two transactions that access the same database items contain their operations in a way that makes the value of some database item incorrect, then the lost update problem occurs.

- If two transactions T1 and T2 read a record and then update it, then the effect of updating of the first record will be overwritten by the second update.

Example:

Transaction-X	Time	Transaction-Y
——	t1	——
Read A	t2	——
——	t3	Read A
Update A	t4	——
——	t5	Update A
——	t6	——

Here,

- At time t2, transaction-X reads A's value.
- At time t3, Transaction-Y reads A's value.
- At time t4, Transactions-X writes A's value on the basis of the value seen at time t2.
- At time t5, Transactions-Y writes A's value on the basis of the value seen at time t3.
- So at time T5, the update of Transaction-X is lost because Transaction y overwrites it without looking at its current value.
- Such type of problem is known as Lost Update Problem as update made by one transaction is lost here.

2. Dirty Read

- The dirty read occurs in the case when one transaction updates an item of the database, and then the transaction fails for some reason. The updated database item is accessed by another transaction before it is changed back to the original value.
- A transaction T1 updates a record which is read by T2. If T1 aborts then T2 now has values which have never formed part of the stable database.

Example:

Transaction-X	Time	Transaction-Y
——	t1	——
——	t2	Update A
Read A	t3	——
——	t4	Rollback
——	t5	——

- At time t2, transaction-Y writes A's value.
- At time t3, Transaction-X reads A's value.
- At time t4, Transactions-Y rollbacks. So, it changes A's value back to that of prior to t1.
- So, Transaction-X now contains a value which has never become part of the stable database.
- Such type of problem is known as Dirty Read Problem, as one transaction reads a dirty value which has not been committed.

3. Inconsistent Retrievals Problem

- Inconsistent Retrievals Problem is also known as unrepeatable read. When a transaction calculates some summary function over a set of data while the other transactions are updating the data, then the Inconsistent Retrievals Problem occurs.
- A transaction T1 reads a record and then does some other processing during which the transaction T2 updates the record. Now when the transaction T1 reads the record, then the new value will be inconsistent with the previous value.

Example:

Suppose two transactions operate on three accounts.

Account-1	Account-2	Account-3
Balance = 200	Balance = 250	Balance = 150

Transaction-X	Time	Transaction-Y
——	t1	——
Read Balance of Acc-1 sum <-- 200 Read Balance of Acc-2	t2	——
Sum <-- Sum + 250 = 450	t3	——
——	t4	Read Balance of Acc-3
——	t5	Update Balance of Acc-3 150 --> 150 - 50 --> 100
——	t6	Read Balance of Acc-1
——	t7	Update Balance of Acc-1 200 --> 200 + 50 --> 250
Read Balance of Acc-3	t8	COMMIT
Sum <-- Sum + 250 = 550	t9	——

- Transaction-X is doing the sum of all balance while transaction-Y is transferring an amount 50 from Account-1 to Account-3.
- Here, transaction-X produces the result of 550 which is incorrect. If we write this produced result in the database, the database will become an inconsistent state because the actual sum is 600.
- Here, transaction-X has seen an inconsistent state of the database.

Concurrency Control Protocol

Concurrency control protocols ensure atomicity, isolation, and serializability of concurrent transactions. The concurrency control protocol can be divided into three categories:

1. Lock based protocol
2. Time-stamp protocol

3. Validation based protocol

Implementation of Locking in DBMS

Locking protocols are used in database management systems as a means of concurrency control. Multiple transactions may request a lock on a data item simultaneously. Hence, we require a mechanism to manage the locking requests made by transactions. Such a mechanism is called as **Lock Manager**. It relies on the process of message passing where transactions and lock manager exchange messages to handle the locking and unlocking of data items.

Data structure used in Lock Manager –

The data structure required for implementation of locking is called as **Lock table**.

1. It is a hash table where name of data items are used as hashing index.
2. Each locked data item has a linked list associated with it.
3. Every node in the linked list represents the transaction which requested for lock, mode of lock requested (mutual/exclusive) and current status of the request (granted/waiting).
4. Every new lock request for the data item will be added in the end of linked list as a new node.
5. Collisions in hash table are handled by technique of separate chaining.

Database security →

Database security is an important issue to keep the database informations correct & prevent it from any unwanted changes.

→ Security policies :-

(a) Identification
(b) Authorization →
(c) Authentication →

(a) Identification →

Identification is the process of identifying the user. If the user is a legitimate person than he is allowed as a genuin person. In database to identify the user, user given login and password.

(b) Authorization →

Authorization is the process which is manage by the database manager or database administrators (DBA). The DBA obtains information about the current authenticated. The DBA then authorizes the authenticated user that which database operations the user can perform or access.