Chap : 1 Database System Architecture

☆ Introduction to Data base Managment system.

→ A Database Managment system is a collection of inter-related data and set of programs to manipulate those data.

DBMS = Database + Set of programs.

→ The Primary goal of DBMS is provide a way to store and retrive the required information from the Database in convenient manner.

→ For Managing data in database two tasks conduct.
(i) Define the structure for storage information
(ii) Provide mechanism for manupulation of information

→ In addition database systems must ensure the safety information being stored.

* Database System Applications :

→ Accounting : Database systems are used to store and maintaining information employees, salaries and Payroll taxes.

→ Manufacturing : For managment of supply chain and tracking production of item in factories

→ Banking : In banking sector DBMS used for managing customer information, accounts and loan.

→ University : The DBMS used in university for store student information, course registration and accounting.

→ Reservation System : The DBMS is used to maintain the reservation and ~~sh~~ schedule information.

→ For Maintaining customer product and purchase information the DBMS are used.

★ Purpose of Database System

→ Earlier Database system are created to manage commercial data. This data typically stored in files. To allow user to manipulate these files, various programs are written for;

(i) Addition of New data
(ii) Updating the data
(iii) Deleting the data.

☆ Explain advantages of DBMS over File Management

→ Data Redundancy : Due to centralized database (Duplication) it is possible to avoid duplication of information.

- This lead to reduce data redudancy.
- It prevent memory wastage.
- It also reduce extra processing time to get required data.

→ Shared Data : All authorized user and application program can share database ~~esi~~ easily.

→ Data Consistency : Data consistency occurs due to data redudancy.

- With reduce data redudancy the data consistency can be eliminated.
- This results in improved data consistency.

→ Data Access : DBMS utilizes variety of techniques to retrive data.

- Required data can be retrived by providing appropriate query to DBMS.
- Thus data can be access in convenient and efficient ~~mann~~ manner.

→ Data Integrity : Data in database must be correct and consistent.

- So, data stored in database must satisfy certain rules.
- DBMS provide different way to implement such type of rules.
- This improve data intergrity in database.

→ Data Security : Database should be accesible to user in limited way.
- DBMS provide way to control the access to data for different user according to their requirment.
- It prevents unauthorized access data.
- Thus security can be improved.

→ ~~Current~~

→ Concurrent Access : Multiple user are allow to access data simultaneously.
- Concurrent access to centralized data can be allowed under some supervision.
- This results in better performance and faster response.

→ Guaranteed Atomicity : Any operation in database must atomic.
- This means operation can be executed either 100% or 0%.
- This type of atomicity guaranteed in DBMS.

DATE: PAGE:

Advantages of Database System

→ DBMS removes the Data Redudancy that means no duplication of data in database.

→ DBMS allows to retrieve the desired data in required format.

→ Data can be isolated in seprate table for convenient and efficient use.

→ Data can be accessed efficiently using a simple query language.

→ The Data integrity can be maintained. That means the constraint can be applied on data and it should be in some specific range.

→ The atomcity of data can be maintained. That means if some operation is perfomed in particular table of database. then the change must be reflected for the entire database.

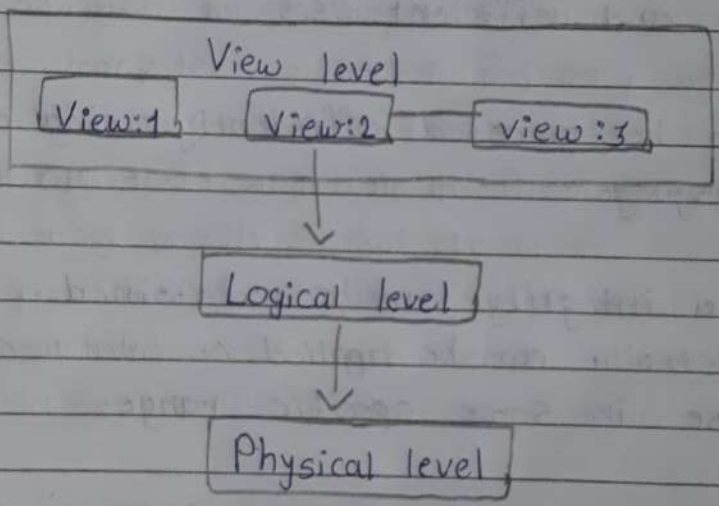
→ The DBMS allows concurrent acess to multiple user by using synchronization technique.

→ The Sequrity policies can be applied to DBMS to allow the user to access only required part of database system.

Data Abstraction

→ Definition : Data abstraction means retrieving only required information of the system and hiding background details.

→ The Data Abstraction divided into three levels:
(i) Physical level
(ii) Logical level
(iii) view level



→ (i) Physical level : This is the lowest level
→ This level describes how actually the data are stored.
→ The database administration decide how to store data in physical level
→ This level describes complex low level data structures.
→ It is also called Internal level.

-> (ii) Logical level : This is the next higher level of the data abstraction.

-> Which describes what data are stored in database.

-> This level also describe the relationship among the data.

-> It describes all records and relationship in simple way.

-> It is also called conceptual level.

-> (iii) View level : This is the highest level of data abstraction.

-> It can provide the access of only part those user required.

-> It helps in simplyfying the interaction with system.

-> It can provide multiple view of the same system.

-> It includes entities, attributes and relationship.

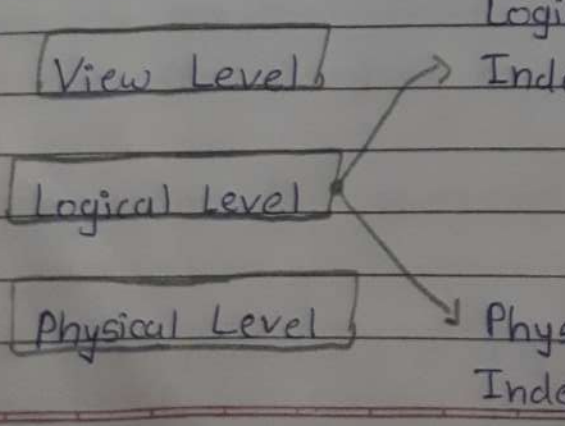
DATE: PAGE:

☆ Data Independence

→ Data independence is ability by which one can change the data at one level without affecting the data another level.
→ Here level can be Physical, Logical and View.
→ There are two types of data Independence.

(i) Physical Independence: This type of data independence which allow modification of physical level without requiring any change to logical level
→ Example: If any change in memory size of database server then it will not affected to logical structure of any data object

(ii) Logical Independence: This is a kind of data independence which allow the modification of logical level without any requiring to change view level.
→ Example: Any change in the table structure such as addition or deletion of some column does not affect user view.



Instance And Schema

→ Schema : The overall design of the database is called schema.

Ex. In a program we do variable declaration and assignment of values to the variables.

→ The variable declaration is called schema and value assigned to the variable is called Instances.

→ The schema for student record can be

Roll No.	Name	Marks

→ Intances : When information is inserted or deleted from the database then database gets changed. The collection of Information at particular moment is called Instances.

→ Student Database

Roll No.	Name	Marks
10	AA	43
20	BB	67

* Types of Schema

(i) Physical Schema : It is describe physical level of abstraction.
(ii) Logical Schema : It is describe Logical level of abstraction.
(iii) SubSchema : A database may have serval views at this view level which are called subschema.

☆ Data Definition Language (DDL)

→ DDL is a specialized language used to specify a database schema by set of definition.
→ It is a language which is used for creating and modifying the structure of tables, views and Index so on.
→ DDL is also used to specify additional properties of data.
→ Some of common commands :
CREATE, ALTER, DROP
→ Use of CREATE command is to build a new table.
→ Use of ALTER command is user can add some additional column and drop existing column.
→ Use of DROP command is the user can delete table or view.

☆ Data Manupulation Language (DML)

-> DML is stands for Data Manipulation language.
-> This language enable users to access or manipulate data as organized by appropriate data model.

-> The types of acess are:

- Reterival of Information stored in the database.
- Insertion of new Information into the database.
- Deletion of Information from the database.
- Modification of Information stored in database.

-> There are two types of DML

(i) Procedural DML: Required a user to speify which data are needed and how to get those data.

(ii) Declaritive DML: Required a user to specify what data are needed without specyfying how to get those data.

-> Query is a statment used for requesting the retrieval of Information. This retrival of Information using some specific language is called Query language

☆ Database System Architecture

→ The two important components of database architecture are - Query processor and Storage manager.

* Query processor :

→ The interactive query processor help the system to simplify and help access to the data.
→ It consists of DDL Interpreter, DML compiler and evaluation engine.

(i) DDL Interpreter : This is bassically translator which explain the DDL statment in data dictionaries
(ii) DML Compiler : It translate DML statement query language into an evaluation plan. This plan consist instruction of which query evalution engine understand.
(iii) Query evalution Engine : It execute the low-level instructions genrated by DML compiler.

→ When a user query issues query, the parsed query is prensented to a query optimizer which use information about how to data stored to produce a execution plan for the query.
→ An execution plan is a blueprint for evaluting a query.

* Storage Manager:

→ Storage manager is the component of database system that provide interface between the low level data stored in database and the application programs.

(i) Authorization and Integrity Manager: Validate the user who want to access data and test for intergrity constraints.

(ii) Transaction Manager: Ensures the data in element is consistent state of the file system failure and transaction execution process without disagree.

(iii) File Manager: Which Manages allocation of space on disk storage and representation of the information on disk.

(iv) Buffer Manager: Manages the fetching of data from disk storage into main memory. The buffer manager also decide what data catch in main memory.

→ Storage Managment Implements serval data structure.

(i) Data Files: Used for storing database it self.

(ii) Data dictionary: Used for sorting med data, particulary schema for med data.

(iii) Indices: Used to provide fast access to data items present in database.

DATE: PAGE:

Chapter: 2

Relational Model

☆ What is Relational algebra?

-> Relation algebra is a language for expressing relational database query.
-> Relational algebra is a procedural query language

☆ Selection

-> Symbol : σ (sigma)
-> Notation : $\sigma_{(condition)} < Relation >$
-> Operation : Select tuples from a relation that satisfy a given condition.
It is used to select particular tuple from a relation.

Student

RNo	Name	Dept	CPI
101	Ramesh	CE	8
108	Mahesh	EC	6
109	Amit	CE	7
126	Chetan	CI	8

-> Find out all the student of CE department.

$\sigma_{Dept = "CE"}(Student)$

Student

R No.	Name	Dept	CPI
101	Ramesh	CE	8
109	Amit	CE	7

☆ Projection

→ Symbol : Π (Pi)

→ Notation : $\Pi_{(attribute\ set)} <Relation>$

→ Operation : Selects specified attribute of relation.

- It selects particular attributes but all tuples from a relation.

Student

RNO	Name	Dept	CI
107	Ramesh	CE	8
108	Mahesh	EC	6
109	Amit	CE	7
125	Chetan	CI	8
138	Mukesh	ME	7
128	Reeta	EC	6
133	Anita	CE	9

→ Example: List out all the students with their roll no, name and department.

$\Pi_{RNO, Name, Dept}$ (Student)

Ans: Output is above table is as follow.

→ Example: List out all the student of CE department with their Roll no., Name and department Name.

Ans: $\Pi_{RNo, Name, Dept}(\sigma_{Dept = "CE"}(Student))$

Student		
RNo	Name	Dept
101	Ramesh	CE
109	Amit	CE
133	Anita	CE

Division

→ Symbol : (÷)

→ Notation : $R \div S$

→ Operation : The division is a binary relation that is written as $R \div S$

- The results consists of the header of R but not in the header of S, for which it hold that all the tuples in S are presented in R.

Project
Task
Database 1
Database 2

Work	
Student	Task
Shah	Database 1
Shah	Database 2
Shah	Compiler 1
Vyas	Database 1
Vyas	Compiler 1
Patel	Database 1
Patel	Database 2

→ Example : Find out all students having both tasks Database 1 as well as Database 2.

$\prod_{(Student)}(Work) \div \prod_{(Task)}(Project)$

Ans :

Student
Shah
Patel

Cartesian Product

-> Symbol : X (cross)
-> Notation : Relation 1 x Relation 2
-> Operation : Combine information of two relations.
- It will multiply first and second relation to each tuples.
- It is also known as cross product operation.

R	
A	1
B	2
D	3

S	
A	1
D	2
E	3

R x S			
A	1	A	1
A	1	D	2
A	1	E	3
B	2	A	1
B	2	D	2
B	2	E	3
D	3	A	1
D	3	D	2
D	3	E	3

→

Emp

Empid	Emp Name	Dept Name
S01	Manisha	Finance
S02	ANisha	sales
S03	Nisha	Finance

Dept

Dept Name	Manager
Finance	Arun
sales	Rohit
Production	Kishan

→

Emp x Dept

Empid	Emp Name	Emp.dept	Dept.dept	Manager
S01	Manisha	Finance	Finnce	Arun
S01	Manisha	Finance	sales	Rohit
S01	Manisha	Finance	Production	Kishan
S02	Anisha	sales	Finance	Arun
S02	Anisha	sales	Sales	Rohit
S02	Anisha	sales	Production	kishan
S03	Nisha	Finance	Finance	Arun
S03	Nisha	Finance	Sales	Rohit
S03	Nisha	Finance	Production	Kishan

Join

→ Tables for Join

Emp		
Empid	EmpName	DeptName
So1	Manisha	Finance
So2	Anisha	Sales
So3	Nisha	Finance

Dept	
DeptName	Manager
Finance	Arun
Sales	Rohit
Finance	Kishan

* Natural Join Operation (⋈)

→ Symbol : ⋈

→ Notation : Relation 1 ⋈ Relation 2

→ Operation : Natural join will reterive information from multiple relation. It works in three step

(1) It perfoms cartesian product

(2) Then find consistent tuples and inconsistent tuples are deleted.

(3) Then it delet duplicate attributes.

→ Example :

Emp ⋈ Dept

Empid	EmpName	Deptname	Manager
So1	Manisha	Finance	Arun
So2	Anisha	Sales	Rohit
So3	Nisha	Finance	Arun

☆ The Outer Join Operation

→ An outer join does not require each record in the two joined tables to have matching record.

→ In natural join some records are missing. If we want that missing records than we have to use outer join.

→ It can be divided into three formate

(1) Left outer join (⟕)
(2) Right outer join (⟖)
(3) Full outer join (⟗)

College

Name	Id	Department
Manisha	S01	Computer
Anisha	S02	Computer
Nisha	S03	It

Hostel

Name	hostel-Name	Room-No
Anisha	Kaveri hostel	K01
Nisha	Godavari hostel	G01
Isha	Kaveri hostel	K02

* Left Outer Join (⟕)

-> The left outer join retains all the tuples of the left relation even through there is no matching tuples in right relation.

-> For such kind of tuples having no matching, the attributes of right relation will be NULL.

-> Example:

College ⟕ Hostel

Name	Id	Department	Hostel-Name	Room-No
Manisha	S01	computer	NULL	NULL
Anisha	S02	computer	Kaveri hostel	K01
Nisha	S03	I.T	Godavari hostel	G07

* Right Outer Join (⟖)

-> For such kind of tuples no matching, the attributes of left relation will be NULL in resultant relation

College ⟖ Hostel

Name	Id	department	Hostel-Name	Room-No
Anisha	S02	Computer	Kaveri hostel	K01
Nisha	S03	I.T	Godavari hostel	G07
Isha	NULL	NULL	Kaveri Hostel	K02

* Full Outer Join (⟗)

→ The full outer join returns all the tuples of both of the relations. It also put NULL values whenever required.

→ Example:

college ⟗ Hostel

Name	Id.	Department	Hostel-Name	Room-No
Manisha	S01	Computer	NULL	NULL
Anisha	S02	Computer	Kaveri hostel	K01
Nisha	S03	I.T	Godavri hostel	G07
Isha	NULL	NULL	Kaveri hostel	K02

Union

→ Symbol : ∪ (union)
→ Notation : Relation 1 ∪ Relation 2
→ Operation : Selected tuples those are in either or both of the relations.

→

R	
A	1
B	2
D	3
F	4
E	5

S	
A	1
C	2
D	3
E	4

R ∪ S	
A	1
B	2
C	2
D	3
F	4
E	5
E	4

→

Emp	
Id	Name
1	Manisha
2	Anisha
3	Nisha

Cst	
Id	Name
1	Manisha
2	Anisha
4	Isha

Emp ∪ cst	
Id	Name
1	Manisha
2	Anisha
3	Nisha
4	Isha

☆ Intersection

→ Symbol : ∩ (Intersection)
→ Notation : Relation 1 ∩ Relation 2
→ Operation : Select tuples those are in both relations.

→ Example :

R	
A	1
B	2
D	3
F	4
E	5

S	
A	1
C	2
D	3
E	4

R ∩ S	
A	1
D	3

→

Emp	
Id.	Name
1	Manisha
2	Anisha
3	Nisha

Cst	
Id.	Name
1	Manisha
2	Anisha
4	Isha

Emp ∩ cst	
Id	Name
1	Manisha
2	Anisha

Difference

-> Symbol : − (Minus)
-> Notation : Relation 1 − Relation 2
-> Operation : Select tuples those are in left relation but not in right relation

-> Example :

R	
A	1
B	2
D	3
F	4
E	5

S	
A	1
C	2
D	3
E	4

R − S	
B	2
F	4
E	5

->

Emp	
Id	Name
1	Manisha
2	Anisha
3	Nisha

Cst	
Id	Name
1	Manisha
2	Anisha
4	Isha

-> Emp − cst

Id	Name
3	Nisha

-> cst − Emp

Id	Name
4	Isha

☆ Rename

→ Symbol : ρ (rho)

→ Notation : $\rho_A(B)$ Rename Relation B to A.

$\rho_A(x1, x2, \ldots xn)$ Rename Relation B to A and its attributes to x1, x2, ..., xN.

→ Operation : It is used to rename a relation or attributes.

→ Student

RNo	Name	Dept	CPI
101	Ramesh	CE	8
108	Mahesh	EC	6
109	AMit	CE	7
125	Chetan	CI	8
138	Mukesh	ME	7
128	Reeta	EC	6
133	Anita	CE	9

→ Example : Find out highest CPI From student table

$$\Pi_{CPI}(Student) - \Pi_{A.CPI}(\sigma_{A.CPI < B.CPI}(\rho_A(Student) \times \rho_B(Student)))$$

CPI
9

☆ Aggregate Function

→ Symbol : G
→ Notation : G function(attribute) (relation)
→ Operation : It takes a more than one value as input and returns a single value as output.
→ Function : Sum, Count, Max, Min, Avg.

→

Student			
RNo	Name	Dept	CPI
101	Ramesh	CE	8
108	Mahesh	EC	6
109	Amit	CE	7
125	Chetan	GI	8
138	Mukesh	ME	7
128	Reeta	EC	6
133	Anita	CE	9

→ Example : Find out sum of all student CPI.

$G_{sum(CPI)}$ (Student)

Sum
51

→ Example : Find out max and mix CPI

$G_{max(CPI), min(CPI)}$ (Student)

Max	Min
9	6

Que: Consider following schema and represent given statments in relation algebra form.

* Branch (branch-name, branch-city)
* Account (branch-name, acc-no, balance)
* Depositor (customer-name, acc-no)

Ans

→ (i) Find out list of customer who have an account at 'abc' branch.

→ $\Pi_{\text{customer-name}}(\sigma_{\text{branch-name}=\text{"abc"}}(\text{Depositor} \bowtie \text{Account}))$

(ii) Find out all customer who have an account in Ahemedabad and balance is greater than 10lk.

→ $\Pi_{\text{customer-name}}[\sigma_{\text{Branch.branch-city}=\text{"Ahemedabad"}} \wedge \sigma_{\text{Account.balance}>10000}(\text{Branch} \bowtie \text{Accont} \bowtie \text{Depositor})]$

(iii) Find out list of all branch name with their maximum balance.

→

$\Pi_{\text{branch-name}}, G_{\max(\text{balance})}(\text{Account})$

Chapter: 2

Relation Model

Solve the quries for the following database using relational algebra.

~~branch (branch-name, customer-street, customer-only)~~

branch (branch-name, branch-city, assets)

Customer (customer_name, customer-street, customer-only)

account (account-number, branch-name, ammount)

loan (loan-number, branch-name, ammount)

depositor (customer-name, account-number)

borrower (customer-name, loan-number)

(i) Find all loans over $1200.

(ii) Find the loan number for each loan of an ammount greater than $1200.

(iii) Find the names of all customers who have a loan, an account or both from the bank.

(iv) Find names of all customers who have a loan an account at the bank.

(v) Find the names of all customers who have a loan at the perryridge branch.

(vi) Find the name of all customers who have a loan at the perryridge branch but do not have an account at any branch of the bank.

(vii) Find the names of all customers who have a loan and an account at the perryridge branch.

writangel

DATE: PAGE:

Ans.

→ (i) $\sigma_{amount > 1200}(loan)$

(ii) $\Pi_{loan\text{-}number}(\sigma_{amount > 1200}(loan))$

Ex 2

(iii) $\Pi_{customer\text{-}name}(borrower) \cup \Pi_{customer\text{-}name}(depositor)$

(iv) $\Pi_{customer\text{-}name}(borrower) \cap \Pi_{customer\text{-}name}(depositor)$

(v) $\Pi_{customer\text{-}name}(\sigma_{branch\text{-}name = "perryridge"}(\sigma_{borrower.loan\text{-}number = loan.loan\text{-}number}(borrower \times loan)))$

(vi) $\Pi_{customer\text{-}name}(\sigma_{branch\text{-}name = "perryridge"}(\sigma_{borrower.loan\text{-}number = loan.loan\text{-}number}(borrower \times loan)))$

(vii) $\Pi_{customer\text{-}name}(\sigma_{branch\text{-}name = "perryridge"}(\sigma_{borrower.loan\text{-}number = loan.loan\text{-}number}(borrower \times loan))) \cup \Pi_{customer\text{-}name}(depositor)$

.2 Consider the relational database as given below. Give an expression relational algebra to express each of the following quries.

Works (Person-name, company-name, salary)
Employee (Person-name, street, city)
Company (company-name, city)
Manages (Person-name, manager-name)

(a) Find the names of all employee in this database who live in the same city as the company for which they work.

(b) Find the names, street address, and cities of residence of all employees who work for HCL corporation and earn more than $10,000 per anum.

ns.

→ (a) $\Pi_{\text{Person-name}}$ (works ⋈ Employee ⋈ Company)

(b) $\Pi_{\text{person-name, street, city}}$ ($\sigma_{\text{company-name} = \text{"HCL"} \wedge \text{salary} > 10000}$ (works ⋈ Employee))

Ex.3 Consider following schema and represent given statm in relation algebra form

Branch (branch_name , branch_city)
Account (branch_name , acc_no , balance)
Depositor (customer_name , acc_no)

(i) Find out all the list of customer who have account in 'abc' branch.
(ii) Find out all customer who have account in 'Ahemedbad' city and balance is greater than 10,000.
(iii) Find out list of all branch name with their maximum balance.

Ans.

→ (i) $\Pi_{\text{customer\_name}} (\sigma_{\text{branch\_name} = \text{"abc"}} (\text{Account} \bowtie \text{Depositor}))$

(ii) $\Pi_{\text{customer\_name}} (\sigma_{\text{branch\_city} = \text{'Ahemedabad'} \wedge \text{balance} > 10,000} (\text{Branch} \bowtie \text{Account} \bowtie \text{Depositor}))$

(iii) $\text{branch\_name}\ \mathcal{F}_{\max(\text{balance})} (\text{Account})$

4 Consider the following relational data base, where the primary keys are underlined. Give an expression in the relational algebra to express each of the following quries

employee (ssn name, dno, salary, hobby, gender)
department (dno, dname, budget, location, mgrssn)
works-on (ssn, pno)
Project (pno, P.name, budget, location, goal)

(i) List out all pairs of employee names and the project numbers they work on

(ii) List out department number, department name and department budget.

(iii) List all projects that Raj Yadav works on by project name.

(iv) List the name of employee who supervise themselves.

ns

→ (i) $\Pi_{name, pno}$ (employee ⋈ works-on)

(ii) $\Pi_{dno, dname, budget}$ (department)

(iii) Raj Yadav ← $\sigma_{name = "Raj\ Yadav"}$ (employee)

Result ← Π_{pname} (Raj Yadav ⋈ works-on ⋈ Project)

(iv) Insufficient data.

Ex. 5 Consider following relational database, where the primary keys underlined. Give an expression in the relational algebra to express each following queries.

Course(course_id, title, dept-name, credits)
instructor(id, name, dept_name, salary)
Section(course-id, sec-id, semester, year, building, room-no, time-slot-id)
Teacher(id, course-id, sec-id, semster, year)

(i) Find the names of all instructors in the physics department.
(ii) Find the all course taught in the fall 2009 semester but not in spring semester.
(iii) Find the name of all instructor in com.sci.department together with course titles of all courses that the instructor teach.
(iv) Find the average salary in each department.

Ans

→ (i) $\Pi_{name}(\sigma_{dept\_name='physics'}(instructor))$

(ii) $\Pi_{course-id}(\sigma_{semester="Fall" \wedge year=2009}(section)) - \Pi_{course-id}(\sigma_{semester='Sprint'}(section))$

(iii) $\Pi_{name, title}(\sigma_{dept-name="comp.sci.department"}(course \bowtie instructor \bowtie teaches))$

(iv) $dept\text{-}name\ \tilde{\mathcal{F}}\ avg(salary)(instructor)$

Chapter : 3
Entity - Relationship Model

☆ Introduction to Relation Model

→ Relation database is a collection of tables having unique names.

→ Consider the example of student table in which the information about the student is stored.

Roll no	Name	Phone
001	AAA	11111
002	BBB	22222
003	CCC	33333

→ The above table consists of three column headers Roll no, Name and Phone. Each row of the table indicates the information of each student by mean of his Roll no, Name and Phone number.

→ Similarly consider another table,

Course ID	Course Name	Credit
101	Mechanical	4
102	Computer	6
103	Electrical	5
104	Civil	3

→ Clearly, in above table the columns are Course ID, Course Name and Credits.

→ The Course ID 101 is associated with the course named Mechanical and course of mechanical there are 4 credits. Thus the relation represented by the table in the relation model.

→ Similarly, we can establish the relationship among the two tables by defining the third table.

Roll No	Course ID
001	102
002	104
003	101

→ From, this third table we can easily find out that course to which is the Roll No. 001 is addmitted is computer.

→ Table or Relation: In Relation Model, table is a collection of data items arranged in rows and columns. The table can not have duplicate data or rows.

→ Tuple: The single entry in the table is called tuples. The tuple represent set of related data. Tuples is also called row.

→ Attributes: It is a part of table contains serval records. Each record can broken down into serval small parts of data known as attributes. It is also known as column.

→ Relation Schema: A relation schema describe the structure of the relation, with the name of the relation (Name of Table), its attributes and their name and type.

→ Relation Intance: It refers to specific intance of relation that means containing a specific set of rows. For Example - which contains the record with marks above 80.

→ Domain: The values are presented in rows and column that is called domain.

→ Degree: It is nothing but total numer of columns present in the relational database

→ Cardinality: It is nothing but total number of rows present in the relational database

→

Informal Terms		Formal Terms
Table	→	Relation
colums header	→	Attributes
Rows	→	Tuples
Value of row	→	Domain
Table definitor	→	Schema of relation

☆ Super key

→ It is a defined as a set of attributes within a table that can uniquely identify each record within a table.

→ Super key is a super set of candidate key.

Std ID	Name	Phone	Age
1	AAA	1111	17
2	AAA	2222	19
3	BBB	3333	18
4	CCC	3444	19

→ Std ID is uniquely in every row of data hence it can be used to identify each row uniquely.

→ (Std Name, Std ID) name of two student can be same but their ID can not be same hence this combination is also key.

→ Phone number is unique, hence it is also key.

Candidate Key

→ Candidate key is a super key without redudancy.
→ It is not reducible further.
→ Minimum and Minimal set of attribute use it to uniquely differentiate record of the table.
→ It can be single and combination of mim minimal attributes.

Table : 1

S.ID	S.Name	Marks
S_1	A	40
S_2	A	40
S_3	B	50
S_2	B	50

Table : 2

S.ID	S.Name	Marks
S_1	A	40
S_2	B	20
S_3	A	20
S_4	C	30

→ In table 1 if we consider S.ID and S.Name there is a no reduddance data that's why it it called combination of candidate key.

→ In table 2 if we consider S.ID there is a no redundance data and it is uniquely that's why it is called candidate key.
→ A candidate key never be null or Empty.
→ And It's value should be unique.
→ There can be more than one candidate key.

Primary Key

→ The Primary key is a candidate key choosen by the database designer to identify the tuples uniquly.

Std ID	Name	Phone	age
1	AAA	1111	16
2	BBB	2222	17
3	AAA	3333	18
4	CCC	4444	17

→ We can consider Std ID as a Primary key.
→ Unique values must be never null.
→ Uniquely identify each record in table.
→ Primary key is a mandatory.
→ Table each record must have a value.
→ When choosing a primary key from the pool of candidate key alway choose a single and simple key over composite key.

Foreign key

→ Foreign key is a single attribute or collection of attributes in one table that refers to the primary key of other table.

→ Foreign keys refer to primary key.

→ The table containing the primary key is called parent table and the table containing a foreign key is called child mode.

→

Reg No	Roll no	Phone	Name	Marks
R101	001	111 AAA	AAA	83
R102	002	222	BBB	88
R103	003	333	CCC	93
R104	004	444	DDD	67

→

Course ID	CourseName	Reg No.
C111	Computer	R103
C112	Electrical	R101
C113	Mechanical	R104
C114	Civil	R102

→ From above examples, we can see that two tables are linked. Find out the student 'CCC' has opted for computer course.

☆ Integrity Constraint

→ Constraint means some rules or restrictions that are set on the database.
→ Intigrity constraints are rules that are to be applied on database column to ensure the validity of data.
→ Integrity constraint ensure that the data insertion, updating and other process have to be perfomed in such way integrity constraint not affected.
→ There are four type of constraint.
(i) Key constraint
(ii) Referal Constraint
(iii) Entity Integrity constraint
(iv) Domain constraint.

(iii) Entity Integrity ~~Rel~~ Constraint:

→ In this relation value of attribute key of primary key can not be NULL.
→ The NULL Represent a value of attribute currently unknown.
→ The NULL is always deal with incomplete data.
→ The primary key help in uniquely identifying every row in the table. If the user of the database retrieve any row from table and perfome any action they must know value of the key for that row.

(ii)

Prima
key

Roll No.	Name	Marks
1	AA	86
2	BB	87
3	CC	88
	DD	89

↓

[NULL is not allowed !!!]

(ii) Referential Integrity constraint

→ In this relationship dat linked with two or more table.
→ It is perfome by foreign key reference Primary key.
→ Whenever a foreign key value is used it must reference a Valide Primary key in Parent table.

Primary key → Reg No	Roll No.	Name
R101	001	AAA
R102	002	BBB

Foreign Key

Course Name	Reg No
Computer	R101
Mechanical	R102
Electrical	R555 ← This value not present Parent table

→ R55 is not existing in Parent table still present in Child table that it is not following referential integrity constraint

(i) Key Constraint

→ Keys are used to identify particular record from table.
→ Primary key normally used to identify the record uniquely.
→ key constraint can be stated as :
- All value of Primary key must be unique.
- The values of Primary key not be NULL.

→ Consider student table Roll No is a primary key. We can find desire record using primary key.
→ Primary key does not have same values.

(ii) Domain Constraint

→ Domain constraint defines the set of values for an attribute.
→ The domains includes string, char, int, time, date, etc.

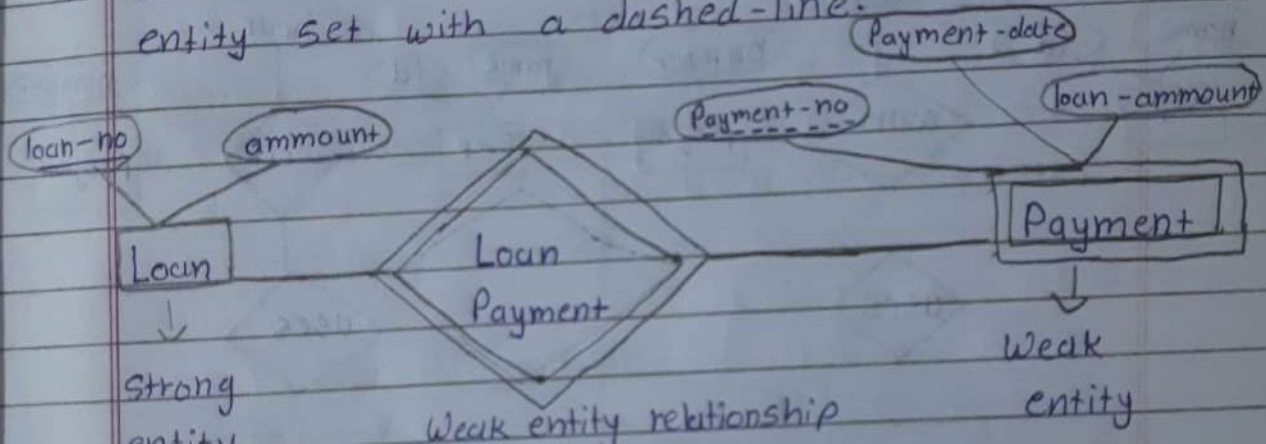
Roll No	Name	Marks
001	AAA	87
002	123	89

Name can not be numeric
It must be string

→ The above relation does not satisfy domain

☆ Weak entity Set

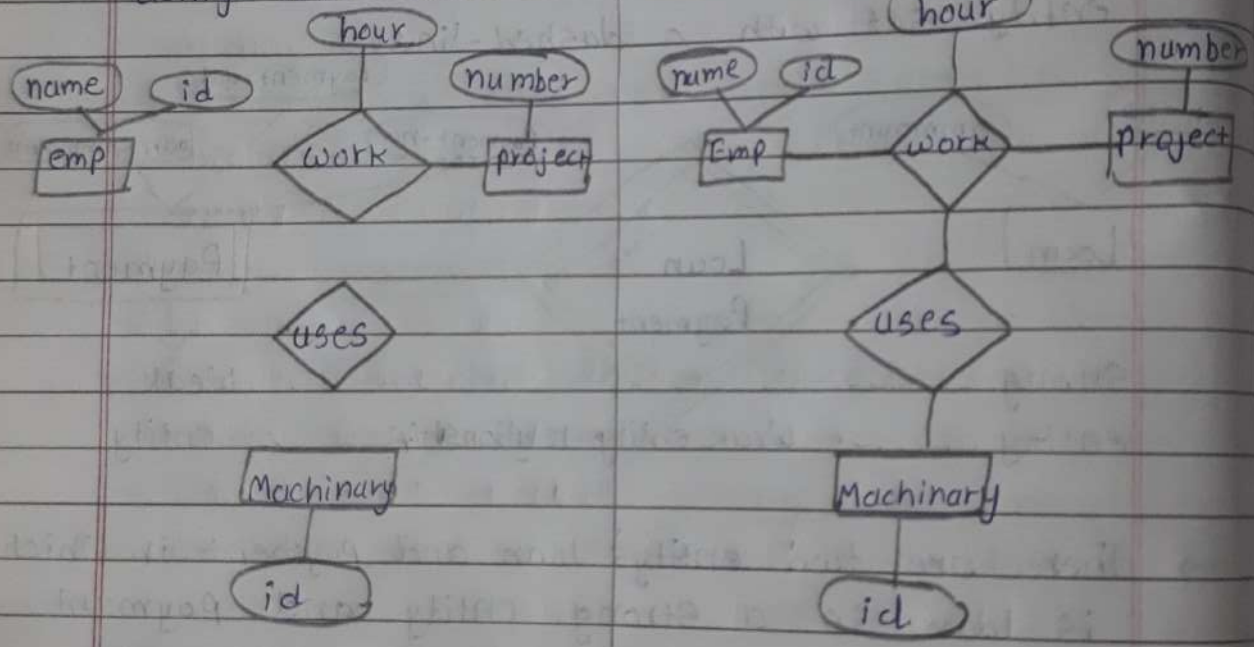
→ An entity set that does not have primary key associated with is called weak entity set.
→ The existance of a weak enitity set depends on existance of strong entity set.
→ Weak entity set indicated by double ~~diamond~~ rectangle.
→ Weak entity relationship set indicates by double diamond.
→ We underline discriminator attributes of weak entity set with a dashed-line.



→ There are two entitys loan and payment in which is loan is a strong entity and payment is a weak entity set.
→ Payment entity has payment-no which is discriminator.
→ Loan entity has loan-no as a primary key.
→ So primary key for payment is (loan-no, Payment-no).

☆ Explain aggregation in E-R Diagram.

→ The E-R Model can not express relationships among relationships.
→ When would we need such a thing at that time aggregation is used.
→ Consider a database with information about employees who work on particular project and use a number of machine doing that work.



→ Relationship sets work and uses could be combined into single set. We can combine them using aggregation.
→ For our Ex. We treat the relationship set work and the entity sets employee and project as higher-level entity set called work.
→ The table for relationship set contains a column for each attribute in the primary key of machinary and work.

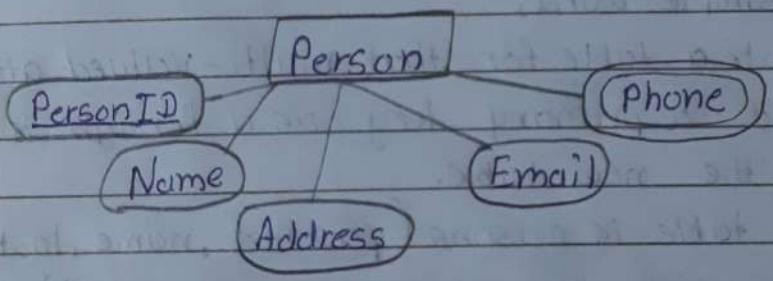
☆ Reduce E-R diagram to E-R Database Schema

Step:1 Simple Attributes (Entity):

-> A entity type within E-R diagram is turned into a table. You may keep the same name for entity or give it a name but avoid DBMS reserved word as well as avoid the use of special character.

-> Each attributes turns into a column in the table. The key attribute of the entity is the primary key of the table which is underline.

-> It is highly recommended that every table should start with primary key attribute conventially named as TablenameId.



-> The relation schema exprssed in following format

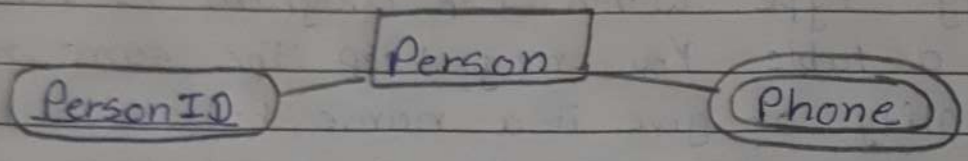
- Person (PersonID, name, address, email)

-> Persons and Phones are Tables and personid, name, address and emails are colums.

-> Personid is a primary key for the table: Person.

Step:2 Muti-Valued Attributes

-> A multi-valued attribute is ussaly represent with double-line.



-> If you have multi-valued attribute take that multi-valued attribute and turn into a new entity or its own table.

-> Then make 1:N Relationship between the new entity or existing one.

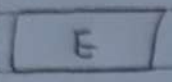
-> In simple word,
- Create a table for that multi-valued attribute.
- Add a primary key as a foreign key in the new table.

-> First table is persons (personid, name, lastname, email)

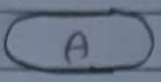
-> Second table is Phones (phoneid, personid, phone)

-> personid ~~whit~~ within the table Phone is a foreign key reffering to the personid of Person.

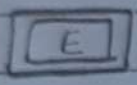
What is E-R Model and draw various Symbol:



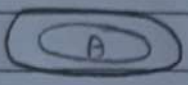
Entity set



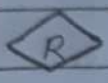
Attribute



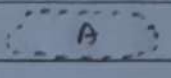
Weak entity set



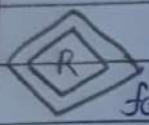
Multivalued att.



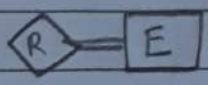
Relationship set



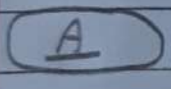
Derived att.



Relationship set for weak entity set

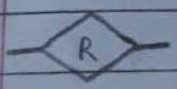


Participant of entity set in relationship



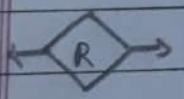
Primary key

Discriminating att. of weak entity set

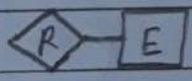


Many to Many Relationship

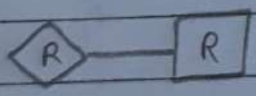
Many to one Relationship



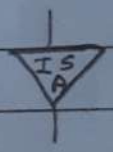
one to one relationship



Cardinality limits



Role Indicator



ISA (Specialization or generalization)

Total Genearealization

Disjoint Genrealizatio

Chapter : 4

Relational Database Design

☆ Function Dependency

→ This is a relationship that exists when one attribute uniquely determines another attribute.
→ Let attributes x and y are two subsets of relation R.
→ If the value of x component of the tuple uniquely determine the value of the Y component, there is a function dependency from x to y.
→ This is denoted by $X \rightarrow Y$
→ Y is functionally dependent on x and x is functionally determines Y.
→ The set of att. x is called left hand side F.D and att. Y is called right side F.D.
→ The left side F.D is also called as determinate and right side F.D is called as dependent.

X	Y
Determinate	Dependent

Ex.

ID	Name	Surname
S_1	AAA	C
S_2	BBB	D
S_3	AAA	E

→ ID → {Name, Surname}

☆ Full FD

→ ABC → D

→ D is fully functional dependent on ABC.
→ D can not depends on any individual any Sbuset ABC.

→ BC → D (BC can not determine D)
C → D (C can not determine D)
A → D (A can not determine D)
Only ABC determines D. that means D is fully functional depentent on ABC.

Ex.

Enrollment_No.	Departmant_Name	SPI
111	AAA	8
222	BBB	9
333	CCC	7

→ {Enrollment_No, Departmant_Name} → SPI

→ That means SPI is FD on Enrollment-No and Departmant-name.

writangel
DATE: PAGE:

☆ Transitive Dependency

→ In a relation, if attributes A → B and B → C then C is transitively depends on A via B.

Ex.

(A)	(B)	(C)
EmpNum	DeptNum	DeptName

→ EmpNum (A) → DeptNum (B) A → B
DeptNum (B) → DeptName (C) B → C

→ DepartmantName (c) transitively depends on EmpNum (A) via DeptNum (B).

EmpNum (A) → DeptName (C) A → C

Ex.

(A)	(B)	(C)
Principal	Teacher	Student

→ Principal → Teacher A → B
Teacher → Students B → C

→ Student transitively depends on principal.

Students → Principal A → C

☆ Trival FD : X → Y is a trival FD if y is ~~not~~ a subset of X

☆ Non-Trival FD : X → Y is a non-trival FD if y is not a subset of X.

Decomposition

→ It is Eliminate reludancy by decomposing a relation into seprate relation in a higher norm fo

→ Let R be a relation schema a set of relation schema $(R_1, R_2, \ldots, R_n)$ is a decomposition of R if $[R = R_1 \cup R_2 \cup R_3 \cup \ldots \cup R_n]$

→ Relation R(A, B, C, D) there can be two subset of $R_1(A, B, C)$ and $R_2(C, D)$
we get, $R = R_1 \cup R_2$

Example: Account-Branch

Ano	Balance	Bname	Baddress
A01	5000	Vvn	Motabazar
A02	6000	Ksad	Chotabazar
A03	7000	Vvn	Motabazar
A04	8000	Ksad	Chotabazar.

Ans

→ This relation can be divided into two different relations.
(i) Account (Ano, Balance, Bname.)
(ii) Branch (Bname, Baddress.)

Accont

Ano	Balance	Bname
A01	5000	Vvn
A02	6000	ksad
A03	7000	Vvn
A04	8000	Ksad

Branch

Bname	Baddress
Vvn	Motabazar
ksad	chotabazar

→ Account ∪ Branch → Account_Branch

Lossy Decomposition

-> The decomposition of relatio R into R_1 and R_2 is lossy when the join of R_1 and R_2 does not the same relation as in R.

-> This is also called as a lossy-join decomposition.

-> The disadvantage of such kind of ~~information~~ decomposition is that some information is lost during the join to get original relation.

Ex. Account

Ano	Balance	Bname
A01	500	Vvn
A02	500	ksan

-> Decomposition : (R_1 and R_2)

Account-Bal

Ano	Balance
A01	500
A02	500

Bal-Branch

Balance	Bname
500	Vvn
500	Ksan

-> Join : ($R_1 \cup R_2$)

Account-Joined

Ano	Balance	Bname
A01	500	Vvn
A01	500	ksad
A02	500	Vvn
A02	500	ksad

-> We did not get original relation when perfome Join operation this is called lossy decomposition.

⟹ Lossless (Non-loss) Decomposition

→ The decomposition of relation R into R_1 and R_2 is lossless when the join R_1 and R_2 produce the same relation as in R.
→ This is also called as a non-additive decomposion.
→ All decompositions must be lossless.

⟶ Account

Ano	Balance	Bname
A01	5000	Vvn
A02	5000	ksad

↓ decomposition

R_1

→ Acc-Bal

Ano	Balance
A01	5000
A02	5000

R_2 Acc-Branch

Ano	Bname
A01	Vvn
A02	Ksad

↓

$R_1 \cup R_2$ → Join

⟶ Acc-Joined

Ano	Balance	Bname
A01	5000	Vvn
A02	5000	Ksad

Normalization

→ Normalization is the process by which we decompose or divide any relation into one or more than one relation to remove redundant data from your table.

→ It is a step by step process and each step is known as a Normal Forms. It is Reversible process.

→ Need of Normalization :

- Remove reduldant data
- Reduce chance of data error
- Reduces Disk space.
- Improve data intigrity and data consistency.

→ Types :

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce cod Normal Form (BCNF)
- Fourth Normal Form (4NF) → Multivalued dependency
- Fifth Normal Form (5NF) → Join dependecy.

First Normal Form (1NF)

→ A relation is in 1NF Domain of each attribute contains only atomic values.
→ It does not contain any composite or muti-valued attributes or their combinations.

Example:

Cid	Name	Address	contact-no
1	X	Mahasukh soc. Ahm.	1234, 8912
2	y	Nayannagar. Gand.	1374
3	z	Subh flat. Gand.	2389

Problem: Suppose We want to find all the customer for particular city then it is difficult to retrieve.

Sol^n:
→ Insert seprate attributes for each sub att. of composite attribute.
→ Insert seprate attribute for multivalued att. and insert only one value on one attribute and other in other attribute.

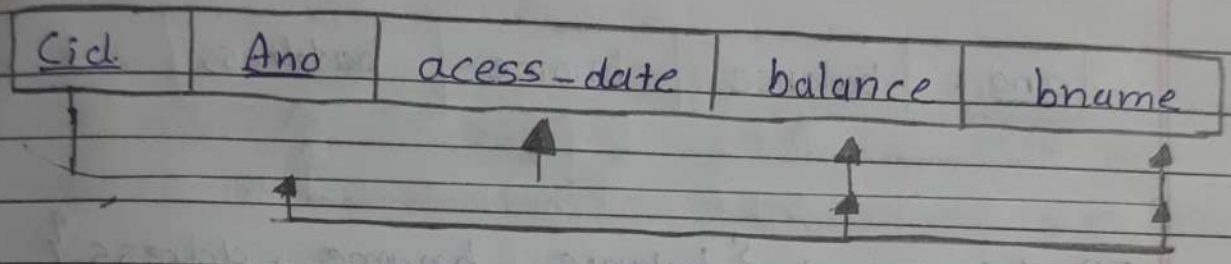
Cid	Name	Society	city	Numer-1	Number-2
1	X	Mahasukh	Ahemeda.	1234.	8912
2	y	Nayanagar	Gandhi.	1374	
3	z	Subh flat	Gandhi.	2389	

DATE: PAGE:

Second Normal Form (2NF)

→ A relation in second normal form (2NF) if and only if it is in 1NF and every non-key attributes is fully FD on the primary key.

Example:

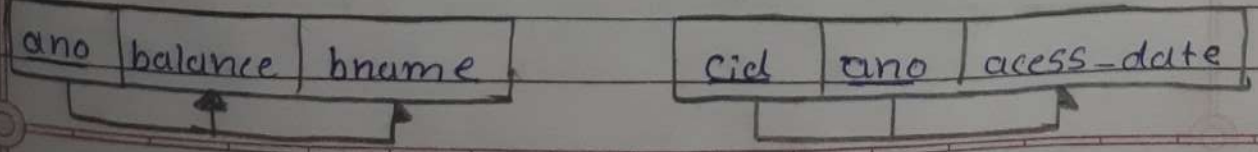


→ FD 1 {Cid, ano} → {acess_date, balance, bname}
FD 2 ano → {balance, bname}

→ We have cid and ano as primary key. As per FD2 balance and bname are only depend on ano not cid. In above table balance and bname are not fully dependent on primary key but these are partial dependent on primary key.

Problem: For example in case of joint account multiple customers have common account.

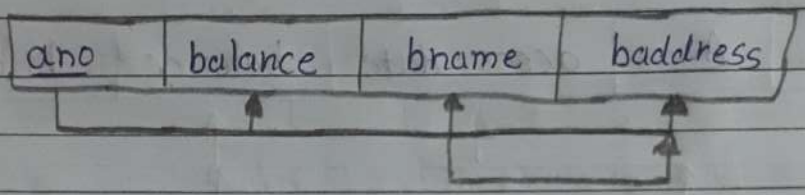
Solⁿ:→ Remove P.D att. that violets 2NF from relation.
→ The prk. of new relation along with the prime att. on which they are fully dependent.
→ The prk. of new relation will be the att. which it if F.D.
→ keep other att. same as in that table with same p.k.



☆ Third Normal Form (3NF)

→ A relation R is in third normal form (3NF) if and only if it is 2NF and every non-key attribute is non-transistively dependent on the P.K.

→ An attribute C is transitively dependent on att A if there exist an attribute B such that:
A → B an B → C



→ FD 1: ano → {balance, bname, address}
FD 2: bname → badress

→ So from FD 1 and FD 2 are using transitively rule we get ano → badress

→ So there is a transitively dependency from ano to badress using bname in which badress is non-prime attribute.

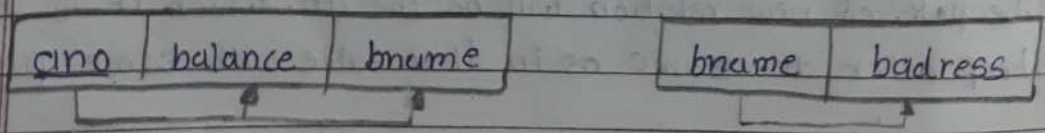
→ So there is non-prime attribute badress which is transitively dependent on primary key ano.

Problem: Transitively dependency resut in data reduclancy.

Sol^{n}: → Remove transitively dependent att that violets 3NF.

→ Place them in seprate new relation among with non-prime att due to which transitive dependency occured.

→ Keep othe att. Same as in the table with same P.K.



☆ Boyce Codd Normal Form (BCNF)

→ BCNF is a strict format of 3NF. A relation is in BCNF if and only if all determinates are candidate keys.

$R(A, B, C) = A \rightarrow B, B \rightarrow C$

A and B are determinate

A and B should be candidate key.

→ Every relation in BCNF is also in 3NF But relation in 3NF is not necessary be In BCNF.

Example

Sid	course	P_name
1	C++	Ashish
2	C	Amit
3	C++	Ashish
4	C	Amit
5	C	Amit
6	C	Amit

→ One student can enroll for multiple courses. For ex. Student with Sid=1 can enroll for C as well as C++.

→ For each course, a teacher is assigned to the student.

→ The candidate key for above table can be (sid, course) because using these two columns we can find.

→ {sid, course} → Teacher
Teacher → course

→ The above table is not in BCNF because of the dependency teacher → course. Note that teacher is not a superkey or in other words teacher is non-prime attribute and course is a prime attribute and non-prime attribute derives the prime attribute.

Soln: Student

Sid	Teacher
1	Ashish
2	Amit
3	Ashish
4	Amit
5	Amit
6	Amit

Course

Teacher	Course
Ashish	C++
Amit	C

Fourth Normal form (4NF)

→ Multi-valued Dependencies : Conditions →

(i) For a dependency A → B, if a single value of A, multiple values of B exists then the table may have multi-value dependency.

(ii) Also table should have at-least 3 columns for it to have a multiValued dependency. between

(iii) And for a relation R(A, B, C), if there is a multi-valued dependency between A and B then B and C should be independent each other.

→ If all these condition are true for Relation It is said to have multi-valued.

→ 4NF : For a table to satisfy the 4NF, it should satisfy the following condition.

(i) It should be in the BCNF.

(ii) And table should not have any multi-valued dependency.

Example Consider following table Relation which is not in 4NF as it contains multi-valued dependency.

Sid	Course	Skill
1	C	English
1	C++	German
1	C	German
1	C++	English
2	Java	English
2	Java	French

→ Student_Course Table

Key : (Sid, Course)

Sid	Course
1	C
1	C++
2	Java

→ Student_Skill Table

Key : (Sid, Skill)

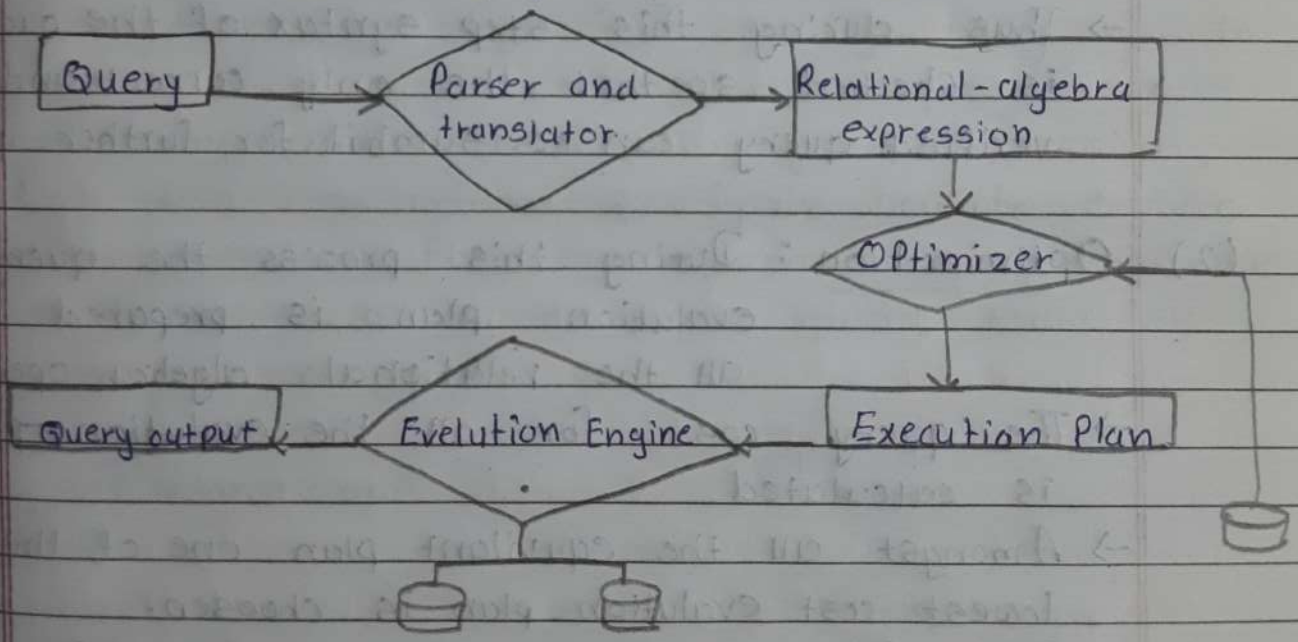
Sid	Skill
1	English
1	German
2	English
2	French

Chapter : 5

Query Processing And Optimization

☆ Basics of query processing

→ Query processing is a collection of activities that are involved in extracting data from database.
→ During query processing there is translation high level dabase language query into the expression that can be used physical level.



→ There are three basic steps in Query processing

(1) Parsing and Translation :
→ In this step the query is translated into internal form and then into relation algebra.
→ Parser check syntax and verify relations.

→ If we sumbit query as,

```
SELECT Roll No., name
FROM Student
HAVING Roll no =10
```

Then it will issues a synthatical error message the correct query should be,

```
SELECT Roll No., name
FROM STudent,
HAVING Roll no =10
```

→ Thus during this step syntax of the query is checked so that the only correct and verified query can be sumbit for further query.

(2) Optimization : During this process the query evalution plan is prepared for all the relational algebra operation.

→ The query cost for all the evalution plan is calculated

→ Amongst all the equvilant plan one of the lowest cost evalution plan is choosen.

→ Cost is estimated using statical information from the database catlog, Such as the number of tuple each relation, size of tuple etc.

(3) Evalution : The query evalution engine takes a query evalution plan, executes that plan and returns the answer to the query.

→ Example:

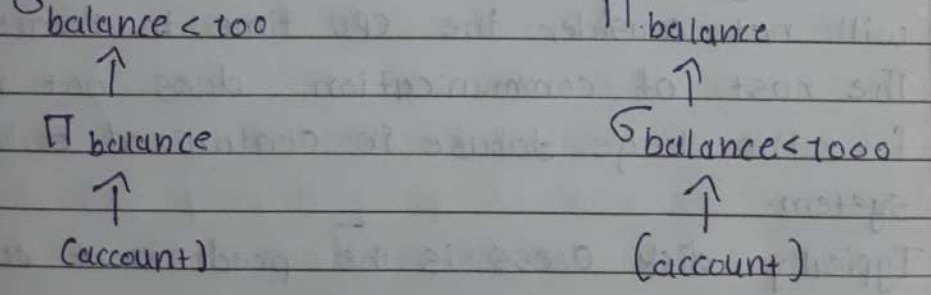
```
SELECT balance
FROM   account
WHERE  balance < 1000
```

Step 1: This query is firstly verify by parser and translator for correct syntax. If so then the relational algebra action can be obtained. Possible relational algebra

(i) $\sigma_{balance<1000}(\Pi_{balance}(account))$

(ii) $\Pi_{balance}(\sigma_{balance<1000}(account))$

Step 2: Query Evalution Plan : To speify fully how to evalute a query we need not only provide relational algebra expression, but also instruction specifying how to evalute each operation. For that purpose using the order of evalution of quries, two query evalution plan is prepared.



→ Each evalution plan there is a query cost. The query optimization selects the query evalution plan having minimum query cost.

→ Once the query plan is choosen, the query is evaluted with that plan and the result of the query is output.

☆ Measuring Cost Of Query

→ There are multiple possible evalution plan for query and it is important to able compare the alternatives in terms of their estimated cost and choose the best plan.

→ There are many factor that contributes to query cost are
- Disk acess
- CPU time to execute the query
- Cost of communication in distributed and parallel database system.

→ The cost of acess data from disk is an import cost. Normally disk acess relatively slow compare to in-memory operation.

→ CPU speed is much faster than the disk spee Hence the time spent in disk is relatively dominated factor in query execution.

→ Computing CPU acess time is harder hence we will not consider the CPU time to execute query.

→ The cost of communication does not matter for the large dabase in centralized database system.

→ Typically disk acess is the predominat cost and is also relatively easy to estimate:
- Number of Seek x Average Seek Cost
- Number of block reads x Average block read cost
- Number of block written x Average block written cost

→ Cost for writen block is greater than the cost to read a block because data is read back after being written.

→ We use number of block tranfered from the disk and number of ~~seek~~ disk seeks to estimate the query cost.

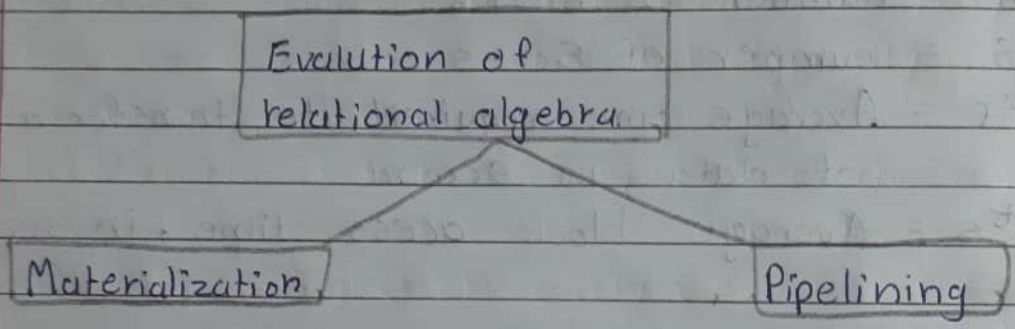
→ Let, Query cost formula = $b * t_T + S * t_S$

- b = number of blocks
- S = number of Seeks
- t_T = Average time required to transfer a block of data, in second
- t_S = Average block acess time, in second.

☆ Evalution of Relational Algebra Expressions

→ As we know query consists of serval relational algebra operators. Evalution of query can be perfomed by evalution of algebraic expressions.

→ Evalution of relational algebra expression can be perfomed using two approaches.



(1) Materialization : The result of relational algebra operator operation is saved in some temporary file which can be used for processing by next subsequent operation. Such an approach is called Materialization.

- This approach is called Materialization approach because - when the result temporary stored in file then it is said that the tuples are Materialized.
- During evalution of query a query tree is built. The process of materialization start at the lowest level operation of the query tree.
- That means the lowest level expression is evaluted, the result is stored in intermediate files, this result is then used by subsequent higher level of the tree, This process repeated and finally root of the query tree giving final result.

- The cost of materialization approach includes the cost of all operation as well as cost of writting result of each intermediate operation to disk.

(2) Pipelining :

- The process of sending the result of one operation to another operation without sorting it to indermediate file is called pipelining.
- This approach improves the perfomance of the query.
- The effciency the query evalution can be improve by reducing the number of temporary files that are produce for storing intermediate result.
- The pipelining approach helps to improve the query evalution process because it does not make use of temporary files and result directly send to the next subsequent operator.

☆ Query Optimization (Heuristic Estimation)

→ Heuristic is a rule that lead to ~~best~~ least cost in most of cases.

→ System may use heuristic to reduce the number of choice that must be made in a cost-based fashion.

→ Heuristic optimization transforms the query-tree by ~~use~~ using set of rules that typically improve execution performance.

(i) Perform selection early (Reduce the number of tuples)

(ii) Perfect Projection early (Reduce the number of attribute).

(iii) Perform most restrictive selection and Join operation before othe similar operations

→ Some System use only heuristic, other combine heurstics with partial cost-based optimization.

⟶ Steps in Heuristic estimation

Step 1: Scanner and parser genrate intial query representation.

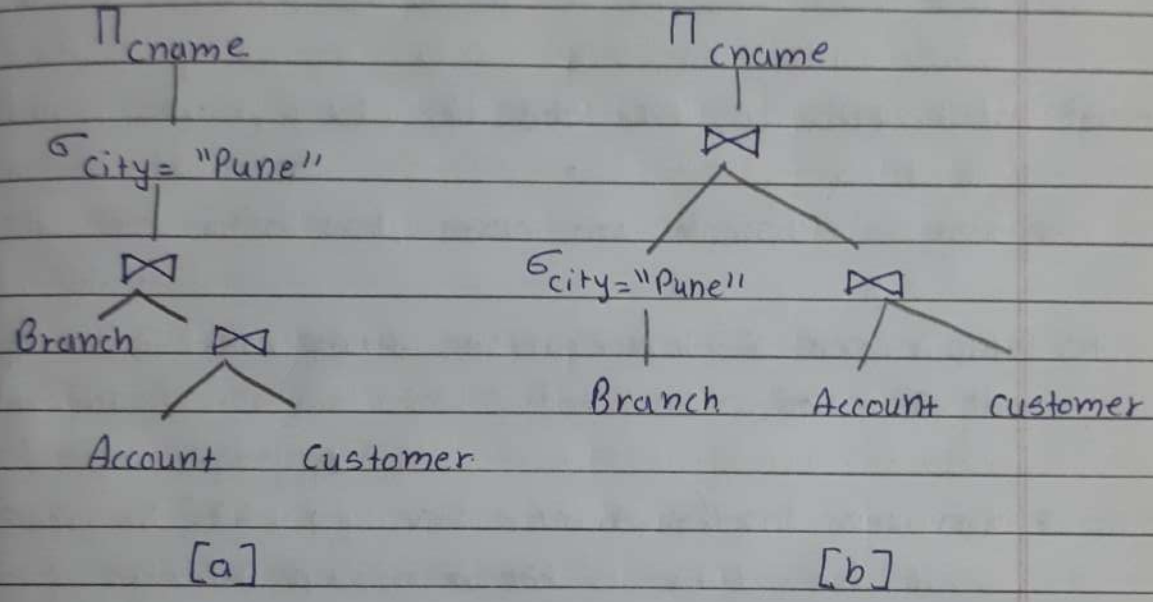
Step 2: Representation is optimized according to ~~the~~ heuristic rules.

Step 3: Query execution plan is devloped.

→ Ex. Suppose there are two relational algebra.

(i) $\sigma_{city = "Pune"}(\Pi_{cname}(Branch \bowtie Account \bowtie customer))$

(ii) $\Pi_{cname}(\sigma_{city = "Pune"}(Branch \bowtie Account \bowtie customer))$



[a] [b]

→ Out of the above query evalution plan the [B] is faster than [A] because in [A] the Join operation among branch, account and customer wherease in [B] the Join of account and customer is made with the selected tuple for city = "Pune". Thus we get choose the optimized query.

Chapter : 7 Security

☆ What is data security

→ Data security is a protection of the data from unathorized user.
→ Only the authorized users are allowed to access the data.
→ Most of the users are allowed to access part of database. i.e. The data that is related to them or their department.
→ Mostly, DBA or head of department can acess all the data in the database.
→ Some user may be permitted only to retrieve data, whereas others are allowed to retrieve as well as update data.
→ The database access is controlled by the DBA.
→ He creates the account of user and gives rights to access the database.
→ User are given usernames protected by password.
→ The user enters his username and password to access the data from database.

☆ Data Encryption

→ Types: (i) Symmetric Encryption (ii) Public key Encryption
→ Encryption is a technique of encoding data, so that only authorized user can understand it.
→ The data encryption technique converts readable data into unreadable data by using some techniques so that unauthorized person can not read it.
→ Ex.

Plain Text → encryption → chiper Text → Decryption → Plain Text

→ In above figure sender having data that he wants to send his data is known as plaintext.
→ In first step sender will encrypt data using encryption algorithm and some key.
→ After encryption the plaintext becomes chipter text
→ The chipertext is not able to read by any unauthorized person.
→ This chipertext is send to reciver.
→ The sender will send that key seprately to reciver.
→ Once reciver recieve chipertext he will decrypt it using that key send by the sender and decrypt the algorithm.
→ After decrypt the algorithm reciver get original data (Plain Text) that is send by sender.
→ This technique is used to protect data when there is a chance of data theft.

☆ Discretionary Access Control (DAC)

→ DAC allows each user or subject to control access to their own data.
→ In DAC, owner of resource restricts access to the resources based on the identify of user.
→ DAC is typically the default access control mechanism for Operating System.
→ Each resource object on DAC based system has Accont Control List (ACL) associated with it.
→ An ACL contains a list of user and groups to which the user has permitted access together with the level of access for each user or group.
→ Ex. test 1.doc : { Prajakta : read }
test 2.exe : {Ankita : Execute}, {Priyanka : execute}
test 3.com : {Ankita : Execute, read },
{Prajakta : Execute, read, write }
→ under DAC user can only set access permission for resources which they already own.
→ User (a) can not change the access control for file that is owned by user (b).
User (a) can set access permision on file that he owned.
→ User can transfer object ownership to another user.
→ User may determine the access type of other user.
→ Advantage : (i) It is flexible. (ii) It is a simple and effcient access right managment.
(iii) It is scalable that means we can add more user.
→ Disadvantage : (i) It increase the risk that data will be made accesible to user that should not neccessarily be given acess.
(ii) There is no control over information one user can transfer ownership to another user.

☆ Mandatory Access Control (MAC)

-> In MAC the access control decision are based on specific between the subject requesting access and object which is requested.
-> MAC genrally used in Government and milatry services.
-> MAC takes a hierarchical approach to controlling access to resources.
-> Under AMC all resource object is controlled and setting by the system administrator.
-> It is not possible to change access control in MAC.
-> In AMC all resource object is assigned by security lable.
(i) Top Secret (TS) (ii) Secret (S)
(iii) Confidential (C) (iv) Unclassified (U)
->(TS) is higher level and (U) is lower level security.
-> Attribute associated with classification
-> In some model Tuple classification (TS) is added.

EmpName	Salary	Perfomance	TS
Ram (U)	500 (C)	Fair (S)	S
Shyam (C)	1000 (S)	Good (C)	S

-> Advantage: MAC provides tighter security because only system administrator may access.
-> Disadvantage: MAC requres careful planning and continuous monitoring to keep all resources object and user classification up to date.

Difference between Authentication and Authorization

Authentication	Authorization
→ It checks identity of user	It checks user access right to access resources.
→ It verify's user ~~credit~~ credential.	It validate's user permission
→ It occures before authorization.	It occurs after authentication.
→ Ex. User can authenticate himself for university exam by login password.	Ex. User can access set of question paper based on access right given him.

Chapter : 9

SQL Concepts

☆ Explain DDL, DML, DCL and DQL

* DDL (Data definition Languages)

→ It is set of SQL commund used to create, Modify and delete database object.
→ It is normally used in DBA and database designer
→ It provides command :
- CREATE = To create object in database
- ALTER = to (alter) modify the existing table
- DROP = delete the object from database
- TRUNCATE = Remove all the record in the table

* DML (Data Manipulation Language)

→ It is set of SQL commands use to insert, Modify, and delete data in database.
→ It Provides command :
- INSERT = to insert data into table
- UPDATE = to modify existing data in table
- DELETE = to delete a record from table
- SELECT = Select a data from table.

* DQL (Data Query Language)

→ It is a component of SQL that allows data reterival from database.
→ SELECT : this command is heart of SQL and allows data to retrieval in different way.

* DCL (Data Control Language)

→ It is set of SQL command is used to controll access data from database.
→ It Provides Command :

- GRANT : to give access privilleges to user on the database
- REVOKE : remove user access rights on the database.

writangel
DATE: PAGE:

* Creation

→ Step 1: We normally create database using SQL

• Syntax :

```
CREATE DATABASE database_name;
```

• Example :

```
CREATE DATABASE Person_DB;
```

→ Step 2 : Table can be created inside the database as follows

• Syntax :

```
CREATE TABLE table_name (
col1_name datatype,
col2_name datatype,
.....
coln_name datatype,
)
```

• Example :

```
CREATE TABLE person_details (
AdharNo. int,
Firstname VARCHAR(20),
Middlename VARCHAR(20),
Lastname VARCHAR(20),
)
```

Output :

Person_details

Adharno	Firstname	Middlename	Lastname

* INSERTION

→ We can Insert data using INSERT statment.

• Syntax :

INSERT INTO table_name ($col_1, col_2, \ldots$)
VALUES ($Value_1, Value_2, \ldots$)

• Ex. :

INSERT INTO person_details (Adhar, Add., City)
VALUES (111, M.G Road, Pune)

O/P Person_details

Adhar	Address	City
111	M.G. Road	Pune

* SELECT

→ It is used to fetch data from database

• Syntax :

SELECT $col_1, col_2, \ldots$ FROM table_name

• Ex :

SELECT Adhar no., First name FROM Person_deta

O/P

Adhar no:	Firstname
111	AAA

→ If we select whole table we make use of (*) character

• Syntax : SELECT * FROM table_name

• Ex : SELECT * FROM Person_details

O/P

Adhar no	Firstname	Middlename	Lastname
111	AAA	BBB	CCC

Table: Person-details

Adhar no.	Firstname	Middlename	Lastname	Address	city
111	AAA	EEE	III	M.G	Pune
222	BBB	FFF	JJJ	Shivaji	Pune
333	CCC	GGG	KKK	Chandni	Delhi
444	DDD	HHH	LLL	Ahemeda	Mumba

* WHERE

→ WHERE command is used to specify some condition.

- Syntax :

```
SELECT col1, col2, ..., coln
FROM table_name
WHERE condition;
```

- Ex.

```
SELECT Adhar no.
FROM Person-table
WHERE city = "Pune";
```

O/P

Adhar no	city
111	Pune
222	Pune

→ Whole table,

- Ex: ~~SELECT * FROM P~~

```
SELECT *
FROM Person-details
WHERE city = "Pune";
```

O/P

Adhar no.	Firstname	Middlename	Lastname	Address	city
111	AAA	EEE	III	M.G	Pune
222	BBB	FFF	JJJ	Shivaji	Pune

* UPDATE

→ It is used to modify existing record of table.

- Syntax :

```
UPDATE table_name
SET col1 = value1
WHERE Condition ;
```

- Example :

```
UPDATE Person_details
SET City = 'Chennai'
WHERE Adhar no = 333 ;
```

O/P

Adharno	Firstname	Middlename	Lastname	Address	city
111	AAA	EEE	III	M.G	Pune
222	BBB	FFF	JJJ	Shivaji	Pune
333	CCC	GGG	KKK	Chandani	Chennai
444	DDD	HHH	LLL	Ahemedab	Mumbai

* DLETION

→ We can delete one or more record based on condition

- Syntax : DELETE FROM table_name
  WHERE Condition ;
- Example : DELETE FROM Person_details
  WHERE Adhar no. = 333 ;

O/P

Adhar no	Firshname	Middlename	Lastname	Address	city
111	AAA	EEE	III	M.G	Pune
222	BBB	FFF	JJJ	Shivaji	Pune
444	DDD	HHH	LLL	Ahemeda	Mumbai

Logical Operators

→ Using where clause we can use the operator such as AND, OR and NOT.

→ AND: Operate displays the record if all condition that are seprated

OR: Operator display the record if any one the condition is seprated

NOT: Operator display the record if condition is not true.

* AND

→ •Syntax:

```
SELECT col1, col2, ...
FROM table-name
WHERE condition 1 AND condition 2 AND ...;
```

•Ex:

```
SELECT Adharno, Firstname, city
FROM Person-details
WHERE Adhar no = 222 AND city = 'Pune';
```

O/P

Adharno	Firstname	city
222	BBB	Pune

* OR

→ • Syntax: SELECT col1, col2, ...
FROM table_name
WHERE Condion 1 OR condition 2 ... ;

• Ex:

```
SELECT AdharNo, Firstname, City
FROM Person_details
WHERE City = 'Pune' OR city = 'Mumbai';
```

O/P

Adharno	Firstname	City
111	AAA	Pune
222	BBB	Pune
444	DDD	Mumbai

* NOT

→ • Syntax: SELECT col1, col2, ...
FROM table_name
WHERE NOT condition;

• Ex.:

```
SELECT Adharno, Firstname, city
FROM Person_table
WHERE NOT city = 'Pune'
```

Adhar no	Firstname	City
333	CCC	Delhi
444	DDD	Mumbai

Order by

→ Many times we need the records in the table to be in sorted order.
→ If the record arrange in increasing order of some column then it is called ascending order.
→ If the record arrange in decreasing order of some column then it is called descending order.
→ The ORDER BY keyword sorts the records in ascending by default.

- Syntax :

```
SELECT col1, col2, ...
FROM table-name
ORDER BY col1, col2... ASC | DESC
```

- Example :

```
SELECT *
FROM Person-details
ORDER BY Adhaarno DESC;
```

O/P

Adhaarno	Firstname	Middlename	Lastname	Address	city
444	DDD	HHH	LLL	Ahemedu	Mumbai
333	CCC	GGG	KKK	Chandani	Delhi
222	BBB	FFF	JJJ	Shivaji	Pune
111	AAA	EEE	III	M.G.	Pune

Alternation

→ It is used to add new column or delete some column from the database.

→ • Syntax for Adding :

```
ALTER TABLE table_name
ADD column_name datatype;
```

• Ex.

```
ALTER TABLE Person_details,
ADD Email VACHAR(30);
```

Adhar no	Firstname	Middlename	Lastname	Add	city	email
111	AAA	EEE	III	M.G	Pune	NULL
222	BBB	FFF	JJJ	Shivaji	Pune	NULL
333	CCC	GGG	KKK	Chandani	Delhi	NULL
444	DDD	HHH	LLL	Ahemedu	Mumbai	NULL

→ • Syntax for deleting:

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

• Ex:

```
ALTER TABLE Person_detail
DROP COLUMN Address;
```

Adharno	Firstname	Middlename	Lastname	city
111	AAA	EEE	III	Pune
222	BBB	FFF	JJJ	Pune
333	CCC	GGG	KKK	Delhi
444	DDD	HHH	LLL	Mumbai

Aggregate Function

→ SQL allows you five built-in aggregate function.
(i) Average : avg (ii) Minimum : min
(iii) Maximum : max (iv) Total : Sum
(v) Count

Ex. Test

id	value
11	100
22	200
33	300
NULL	400

→ Average : SELECT AVG(marks)
FROM Students
→ o/p : 200

→ Count : SELECT COUNT(*)
FROM Test
→ o/p : 4

→ Min : SELECT Min(Value)
FROM Test
→ o/p : 100

→ Max : SELECT Max(Value)
From Test
→ o/p : 400

→ Sum : SELECT Sum(Value)
FROM Test
→ o/p : 1000

☆ Transaction Control Command (TCL)

→ **COMMIT**: The commit command is used to save permanently any transaction to database.

→ When we perform Read or Write operations to the database then those changes can be undone by rollback operation. To make these changes permanant we can use of commit.

→ **ROLL BACK**: It is used to undo transaction that have not already save

→ Student

Roll No.	Name
1	A
2	B
3	C

→
```
DELETE FROM Student
WHERE RollNO = 2;
ROLL BACK;
```

→ O/P:

Student

Roll No	Name
1	A
2	B
3	C

→ SAVEPOINT: It is a point in transaction when you can roll the transaction back to a certain point without rolling backe the entire transaction.

- Syntax: SAVEPOINT savepoint-name;
- ROLLBACK TO SAVPOINT as,

ROLLBACK TO savepoint-name;

Ans

→ Student

Roll NO.	Name
1	A
2	B
3	C
4	D
5	E

→ O/P: Student

Roll NO.	Name
1	A
2	B
3	C

→ Commands:

```
SQL > SAVEPOINT S1
SQL > DELETE FROM Student
      Where Roll no = 2
SQL > SAVEPOINT S2
SQL > DELETE FROM Student
      Where Roll no = 3
SQL > SAVEPOINT S3
SQL > DELETE FROM Student
      where Roll no = 4
SQL > SAVEPOINT S4
SQL > DELETE FROM Student
      where Roll no = 5
SQL > ROLLBACK TO S3;
```

→ O/P

☆ GRANT and INVOKE commands

(1) Grant Command

→ SQL Grant is a command used to provide access on the database object to user.

→ • Syntax:

```
GRANT privilege_name
ON object name
TO {username | Public | role_name}
[WITH GRANT OPTION];
```

→ Privilege_name is the access right granted to the user

→ Object_name is the name of an database object

→ user_name is the name of the user to whom an access right is granted.

→ PUBLIC is used to grant access right to all user.

→ ROLES are a set of privileges grouped together.

→ WITH GRANT OPTION - allow a user to grant access right to other user.

Ex.
```
GRANT SELECT
ON Person_details
To User 1;
```

(2) REVOKE : It is used to remove user access rights to the database object.

→ • Syntax :

```
REVOKE privilege-name
ON object-name
FROM {user-name |PUBLIC |role_name}
```

Ex. To remove access right for SELECT to the table person-details for user 1

```
REVOKE SELECT
ON person-details
FROM user 1;
```

Chapter : 10

PL/SQL Concept

☆ Cursors and its types

→ Cursor is a database object used to traverse the results of a Select SQL query.
→ It is a temporary work area created in the system memory when a select SQL statment is executed.
→ This is temporary work area is used to store the data retrieved from database and manipulate this data.
→ It is point to a certain location within a record set and allow the operator to move forward.
→ We can process only one record at time.
→ The set of rows the cursor hold which is called the active set.
→ Cursor are often and critisized for their higher overhead.

(1) Implict Cursor : These are created by default by ORACLE itself when DML statment like insert, update and delete statment are execute.

- They are also created when SELECT statment that returns just one row execute.
- We can not use implict cursor for user defined work.

(2) Explict Cursor : These are user defined cursor written by the clevloper.

- They can be created when a SELECT statment that return more than one row is executed.
- Even the cursor stores multiple record, only one record can be proceed at a time.
- When you fetch a row the current row position moves to next row.

→ %. FOUND : It will return TRUE, if the DML Statments like INSERT, DELETE, UPDATE and SELECT will affect at least one row else return FALSE.

Ex. SQL %. FOUND

→ %. NOT FOUND : It will return FALSE, if the DML Statment like INSERT, DELETE, UPDATE and SELECT will affect at least one row else return TRUE.

Ex. SQL %. NOTFOUND

→ %. ROW COUNT : Return the number of rown affected by the DML operations INSERT, DELETE, UPDATE, SELECT

Ex. SQL %. ROWCOUNT

→ %. ISOPEN : It will return true if cursor is open else return false.

Ex. SAL %. OPEN

Stored Procedures

Stored procedure is a type of subprogram in PL/SQL block. It is a group of statments that can be called by its name.
This is a subprogram that does not return a value directly.
A procedure is created with the CREATE OR REPLACE PROCEDURE statment.

Syntax :

```
CREATE or REPLACE Procedure Procedure_name
[(Parameter_name [IN|OUT|IN OUT] Type [...])]
[IS|AS]
BEGIN
    Procedure_Body
END;
```

Procedure-Name is used to specify the name of the procedure.
CREATE keyword is used to devlope a new procedure and [OR REPLACE] option allows us to modify an existing procedure.
The optional parameter-list contains name and types of parameter.

→ There are three ways to pass parameters in procedure :

- IN represents that argument valu will be passed from outside the procedure. It is a read only parameter.
- OUT parameter returns a value to the calling program. OUT represents that this parameter will be used to return value outside of the procedure.
- IN OUT parameter passes an initial value to the sub program and returns an updated value to the caller. It is read and write value using this parameter.

→ The Executable part is included in the procedure body.

→ We can execute the process using Execute keyword

→ Syntax :

```
Execute [Procedure_Name];
```

☆ Database Triggers

→ Trigger is something that is invoked automatically when some event occurs.
→ PL/SQL triggers are block structure or pre-defined programs.
→ Trigger is stored in database and invoked repeatedly, when specific condition is match.
→ Triggers are associated with event such as,
- DDL statment such as CREATE, DROP or ALERT
- DML statment such as UPDATE, INSERT or DELETE
- Any other database operation such as Startup, Shutdown, Logging in and Logging out.

→ Syntax:

```
CREATE OR REPLACE TRIGGER trigger-name
BEFORE or AFTER or INSTEAD OF
INSERT or UPDATE or DELETE
of Column-name
ON Table-name
[REFERENCING OLD AS OLD NEW AS NEW]
FOR EACH ROW
WHEN (condition)
DECLARE
    Declaration Section
BEGIN
    Execution section
END;
```

→ • CREATE [OR REPLACE] TRIGGER trigger name: it creates or replace an existing trigger with some trigger-name.

• {BEFORE | AFTER | INSTEAD OF}: when the trigger would be executed the INSTEAD OF clause is use for creating trigger.

• {INSERT | UPDATE | DELETE}: DML operation

• [OF Col-name]: the column name that would be updated

• [ON Table-name]: the name of the table associate with trigger.

• [REFERENCING OLD AS old NEW as new]: refer new and old value for various DML statment.

• [FOR EACH ROW]: It is a row level trigger. the trigger would be executed for each row being affected.

• WHEN (condition): Provides a condition for rows which the trigger would fire. It is valid only for low level triggers.