8012784269 sanjay.

36, 52. 35
14 16
50 45
24. 46
15 56
09. 05
11

RDBMS.

RDBMS - Relational Database management System.

* The software used to store, manage, query and retrieve data stored in a relational database is called RDBMS.

Database:-

* It is an organized collection of data stored and accessed electronically, in a computer system.
* A database is usually controlled by a database management system.
* Small databases can be stored on a file system, while large databases are stored in RDMS.
* Databases make data management easy.

Ex:- Telephone directory store phone numbers and contact details of people.

Types of DBMS:-

1. Relational Databases.
2. network databases.
3. Object-oriented databases.
4. Hierarchical databases.

Erick

(5) E-R model databases.

RDBMS:

* All modern database management systems like SQL, MS SQL Server, MySQL, My SQL, ORACLE, ~~Access~~ are based on RDMS.
* It is called RDMS because it is based on the relational model introduced by E.F Codd
* RDBMS is most commonly used database

How it works?

* Data is represented in terms of tuples (rows) in RDMS
* It contains several tables and each table has its primary key.
* Due to the collection of an organized set of tables, data can be managed & accessed efficiently & easily in RDBMS.
* Everything in a relational database is stored in the form of relations.
* Each table represents some real-world

objects such as person, place, or event about which information is collected.

Example:- Emp details

Columns or Fields / Attributes

primary key

Emp ID	E-name	post	Salary
E1	data · Aadhi value.	manager	80,000
E2			
E3.			

Rows record Tuples

no. of rows = 3.

no. of columns = 4

Emp ID	Years of Exp
E1	15
E2	

Student details

ID	Name	Age	Course.

* Attribute that uniquely identify each row of a table are the primary key.

Relational Database:-

It is a type of database in which the stored data are related to one another.

Database System Applications:-

Databases are widely used. Some are

* Banking:- For customer Info, accounts, loans, and banking transactions
* Airlines:- For Reservations and schedule information
* Universities:- For student info, course registration
* Credit card transactions:- For purchases on credit cards, and generation of monthly statements.
* Telecommunication:- For keeping records of calls made & storing info about the communication networks.
* Sales:- For customer, product & purchase Info.
* Human Resources:- For info about employees, salaries, payroll processing.
* Health care systems:- managing patient records medical billing
* Online Booking systems:- Hotel booking, ticket, !

	RDBMS	DBMS
(1)	Data stored is in table format	Data is in the file format
(2)	multiple data elements are accessible together	Individual access of data elements.
(3)		No connection b/w data.
(4)		small quantity of data stored.
(5)		Data redundancy is common.

DBMS:- is a software package that can be used for creating and managing databases. A RDMS is a database management system that is based on the relational model.

* ORACLE is a specific brand of RDBMS developed by Oracle corporation.
* RDBMS is a type of DBMS that organises data into tables with rows & columns, (insert, update, retrieve data)
* RDBMS uses Sql to interact with the data stored in the tables. (communicate)
* It is the standard language for interacting with the RDBMS, including Oracle.

1. Database - Collection of data is known as database.

2. DBMS → Collection of data and a set of programs to access that data.

* A database is usually controlled by a Database management system.

3. RDBMS:- It is a collection of interrelated data and a set of programs to access that data.

4. ~~Rdbms~~ It is called RDBMS because, it is based on the Relational model introduced by E.F Codd.

3. In Relational model the datas are represented in terms of rows (tuples)

4. In RDBMS the stored data are related to one another.

5. Due to the collection of organized set of tables, data can be managed and accessed efficiently and easily.

Columns / fields / Attributes

	EmpId	E.name	Post	Salary
Rows / Records / Tuples	E1			
	E2			
	E3			

6. Attribute that uniquely identify each row of a table is called primary key.

7. All modern database management systems like SQL Server, ORACLE, My SQL are based on RDMS.

8. RDBMS is the most commonly used database.

Advantages of RDBMS

(1) Simple model: As the structure is simple. It can handled with simple SQL Queries.

(2). Data Accuracy:-
There can be multiple tables related to one another with the use of primary key & foreign key concepts.
Rules are non repetitive. No duplicati of data.

3. Easy Access to Data.
User can effortlessly recover the relevant data as the result.
Select Emp name FROM Employee;

4. Flexibility:-
Can Add the required columns to the table or delete the unwanted columns without affecting the whole structure of the table. No boundaries for e.g. no. of columns, rows or tables.

5. High Security:-
The database can set boundaries by providing access to only particular tables.

Purpose of Database Systems:-
Before database management systems came along, organizations usually stored info in file processing systems.
major disadvantages are:

(1) Data Redundancy and inconsistency.
since diff programmers create files and application programs over a long period.
files -> diff formats
app pgms -> diff prog. Lang.
Same Info can be duplicated in several places (files). Ex: Address & phone no of particular customer may appear in savings account records & checking Account records.
Inconsistency, a changed address may reflect in savings but not in Checking A/C.

Saving A/c
Add phone

Checking A/C
add Phone no.

(2) Difficulty in Accessing Data.
If one bank officer needs names of all customers who lives within a area, there is no application program to hand to retrieve either
- write new application program
- Do it manually.

3. Data isolation:- Because data is scattered in various files, and in diff formats, writing new pgm to access data is difficult.

4. Integrity problems:- (set of rules)
The data values stored in a database must satisfy certain types of consistency constraints. Ex: min balance. Rs.500. Developers enforce them by adding appropriate code in application pgms. When new constraints are added, it is difficult to change the programs to enforce them.

5. Atomicity problems:-
Consider a pgm to transfer Rs 5000, from Account A to Account B. If a system failure occurs during the execution of pgm it is possible to remove Rs 5000 from A but not credited to B.
Atomic → It must happen entirely or not at all.

6. Concurrent Access anomalies:-
Consider Account A → Rs 10,000. If two customers withdraw funds Rs 10,00 Rs 2000 at the same time the result of concurrent executions may leave the account in incorrect state. The Account may contain 8,000 (or) 9000 depends on which write last instead of 7,000

7. Security problems:-
Not every user of the database should be able to access all the data. Ex In banking, payroll personal need only to see info about bank employees, not Customer Accounts.

View of data
* Major purpose of database system is to provide users with an abstract view of the data. (ie) the system hides irrelevant details from user is called data abstraction
Ex: how data are stored and maintained
* Since many database users are not computer literate, developers hide the complexity from users
Physical Level:- Lowest level of Abstraction describes how data is actually stored in the database. Complex data structures details at this level.

Logical Level:-

* middle Level of Abstraction.

It describes what data are stored in the database and what relationships exist among those data are logically implemented

* The programmers generally work at this level. (database Administrator)

View Level:- Highest Level of data Abstraction. This level describes the User interaction with database system.

Ex:- Storing Customer Information in a Customer table.

At physical Level: These records can be described as blocks of storage (bytes, gigabytes, terabytes) in memory. These details often hide from programmers.

At Logical Level:-

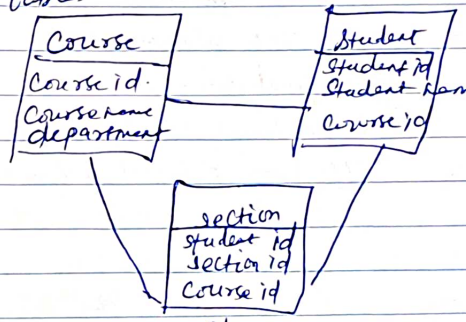
Records can be described as fields and Attributes along with their data types, their relationship among each other. programa

At View level:-

Users

Instance and Schema in DBMS.

Schema:- Design of a database is called schema. only the Structural view of a database. Schema shows relationship between three tables



It does not ~~have~~ show the data present in those tables, shows only the design of the database.

Instance:- The data stored in database at a particular moment of time is called Instance of database.

Ex:- Schema - variable declaration.

Instance - values of the variable at a moment of time.

Ex:- 100 records in a table student today so instance going to be 100.

Add 100 records tommorow. So Instance will be 200 tomorrow.

Data Independence:-

The ability to modify a schema definition in one level without affecting a schema definition in another level.

- physical Data Independence – modify physical schema without affecting Logical schema.
- Logical Data Independence – modify Logical schema without affecting physical schema.

Data models:

→ It define data elements & the relationship b/w the data elements.
→ use to show how data is stored, connected, and accessed in DBMS.
→ Here, we use set of symbols & text to represent the information, so that the members of the Organisation can understand it.

Data models

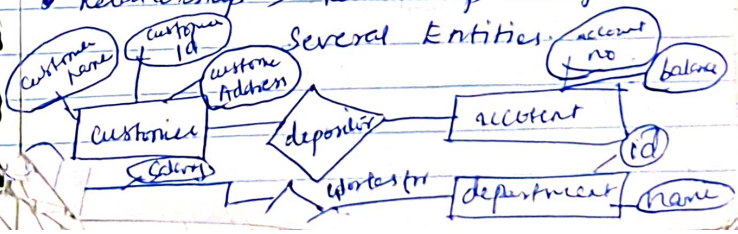
- Object based
  - ER model
  - obj-oriented
- Record based
  - Relational
  - Network
  - Hierarchical.
- physical Data models
  - (i) Unifying model
  - (ii) Frame memory.

① **E-R model.** Anything that has physical existance – Entity

→ Represent the real-world problem in the pictorial form to make it easy for the developers to understand the system, just by looking at the E-R diagram.

E-R diagram has three components

- Entities → Real world thing. It can be Entity a person, place, even a concept. Ex:- Teacher, Student, Course, Customer
- Attributes → the properties of the entity. Ex:- student has property like id, name, age, department.
- Relationship → Relationship among the Several Entities.

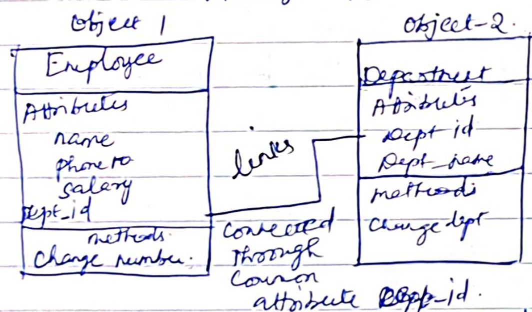


* In this model, the key concepts of OOP languages such as Inheritance, Polymorphism, overloading, Encapsulation and Info hiding can easily be represented.

② Object Oriented model

-> ~~In this model~~ both the data and the relationship are present in the single structure known as object.

-> In this, two or more objects are connected through links.



③ Relational model -> Represents the database as collection of Relations / tables of values

-> model is most widely used, introduced by Dr. E F Codd.

-> represents data and relationships among data by a collection of tables, each of which has no. of columns with unique names & rows.

attributes / fields / columns

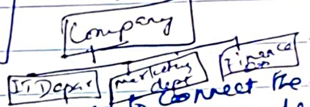
Customer name	customer-id	acc no	customer Address
Tuple / rows records		A-101	
		A-102	
		A-103	

customer database

Account Database => It does not use pointers or links.

=> Relates records by the values they contain

Acc no	Balance
A-101	
A-102	
A-103	



Hierarchical ~~Network~~ model - pointers are used to connect the parent & child node

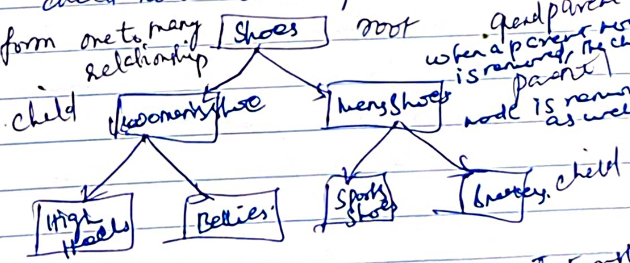
=> this was the first DBMS model

=> It organises the data in the hierarchical tree structure.

1. Any modification to parent is automatically reflected in child node ensuring Data Integrity.

=> The hierarchi starts from the root of and expands in the form of tree, adding child node to the parent node.

-> form one to many relationship



grandparent

when a parent node (i.e.) is removed, the child parent node is removed as well

Network model -> many to many relationship

(graph) => It is an extension of Hierarchical model. It was the most popular model before of Relational model

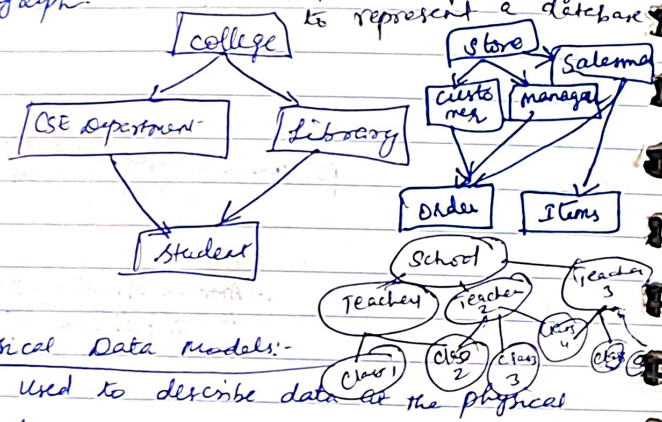
=> The only difference is a record

* Compared to HM, data can be retrieved faster. * Any alteration such as an update, delete, or insertion is difficult.

more than one parent.

⇒ It replaces hierarchical tree with a graph.

* It requires complex diagram to represent a database



Physical Data Models:-

⇒ Used to describe data at the physical level.

⇒ All data ~~are~~ ~~store~~ in a database is stored physically on a secondary storage device such as discs & tapes.

⇒ This is stored in the form of files, other data structures.

⇒ It has all the info of the format in which the files are present and the structure of the database and their relation to each other.

Data Definition Language (DDL) Commands.

1. Create
2. Alter
3. Rename
4. Truncate
5. drop.

↓ used to create and modify the database schema / design / structure of the database.

DML ⇒ used to manipulate the data present in the database.

Create :- To create a new table.

To create a basic table which contains the table name, defining its columns and column's data type, and size of the data.

① Create Database my_database;

② Create Table table_name (column1 datatype(size), column2 datatype(size), ...);

Rollno	Name	Age	Result

Degree ⇒ no of attributes in a relation schema.

Relation Schema ⇒ describes a relation. made up of relation name and a list of attributes $A_1 A_2 \ldots A_n$.

Alter:-

1. Adding a new column.
2. Modify a column
3. Delete a column.

Alter Table Table_name Rename column customer name to Cname;
old-name new-name;

Alter Table table_name Rename To new_table_name;

update table_name set column_name = values.
where Condition;

Select column name from table name.

Unified Data model: brings together data from diff sources and platforms in one place, so that all our data is considered when conducting analyses and making decisions
eg: BI tools, ERP's, product analytics

→ UDM is important because it enables you to look at every collected data point, so you can understand the entire data
Ex: business goals

Desc ⇒ Describe the structure of the table.

Database manager:

⇒ is used to manage all access to the database.
→ Their purpose is to serve as an interface between the database, the user and the applications
⇒ Simply put, Database manager controls any operations executed by the user against database.
⇒ To perform this function, that specific tools are required.

File manager:

is a computer pgm that provides user interface to manage files & folders.

Frame memory:

is designed so that its operating characteristics can be easily manipulated by either designers or design algorithms.

DBA

a person in the organization who controls the design and the use of the database.

1. Schema Definition:-
   - The DBA defines the logical schema of the database.
   - According to this schema, database will be developed to store required data for an organization.
2. Storage Structure & Access method Definition:
   - The DBA decides how the data is to be represented in the stored database.
3. Physical organization modification:-
   - The DBA ~~provides~~ modifies the physical organization of the database to reflect the changing needs of the organization or to improve performance.
4. Granting of authorization for data access:-
   - The DBA determines which user needs access to which part of the database.
   - Acc to this diff type of authorizations are granted to diff users.

DDL commands are (Data Definition Language)

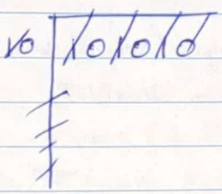
1. Create
2. Alter
3. Rename
4. Truncate
5. Drop

DML Commands are (Data Manipulation Language)

1. Insert
2. Update
3. Delete
4. Select

DCL (Data Control Language)

1. ~~Commit~~ Grant
2. Revoke



* Tablename, columnname can be same, but ~~but~~ two columns cannot have same name, within the same table

Datatypes in Sql.

* char - to store characters - fixed length
* varchar2 → to store (letters, number, special characters) variable length
* number — Integer — number(3)
  float — number(7,2)
* Date Ex:- DOB date

DDL Commands (interact with the database Directly)
↓ Dealing with the structure of the object.

1. Create.

Syntax:-

Create Database databasename;

Eg:-

Create Database IIICSE-2020;
Create Database student;

2. Drop

Drop database student;

1. Table creation:

1. Sql > Create table employee (empid varchar(10), e name Char(20), designation char(15), Salary number(10));
Table created. ⇒

2. Alter table → To modify the structure of the table

a. Add new column :-

Ex: Sql > Alter table employee add age number(2);

Syntax:-

Alter table tablename add columnname datatype(size); ⇒ Sql 11.0

b) Adding more than one column

Alter table tablename add (column_name1 datatype, column_name2 datatype(-));

c) modify the column datatype & size

* alter table employee modify (designation varchar(20));
column name

* delete a column

Alter table <tablename> drop column <column name>;

delete multiple column

(2) alter table tablename drop (col_name);
or
(3) Drop (col1, col2, col3);

Creating Table using Existing Table (with data)

⇒ Sql > Create table Empl as select * from Emp; (with data)

⇒ Sql > Create table Empl as select Eno, Ename, sal from Emp; (with selected columns)

⇒ Create table Empl as select * from Emp where 1=2; (only structure is copied not the data).

DDL - Truncate

Sql > Truncate table <tablename>

Ex:- Truncate table Emp;

Table becomes empty. (we cannot perform roll back)

Rename Table; Oracle 9i (9.2) onwards only.

(1) Rename <oldname> to <newname>;
Rename emp to Employee;

(2) Rename column available in oracle (9.2) only

(1) Alter table <tablename> Rename column <old> to <new>;

Ex: Alter table Emp Rename column Empno to Eno;

Rename Constraint

Alter table <table name> Rename Constraint <old> to <new>;

Ex: Alter table Emp Rename Constraint pk_emp to p_Emp; (pk → primary key)

Dropping constraint: Alter table tablename drop constraint PK;

Drop → Remove the object permenantly from Database.

① Sql> Drop <object> object name;
Drop ≡ Table table name;
Ex: Drop Table Emp;
[If it is log the table will be in Recycle bin].

② Can also drop columns, constraints

Dropping column; (single)

⇒ Alter table <table name> Drop column <column name>;
(or)
Alter table <table name> Drop(col_name);

Dropping multiple columns (multiple)
<Table name>
Alter table Emp Drop(col1, col2, .. coln);

Flash back introduced in log.
Select * from Recyclebin; (Sql Command)
(or)
Show Recyclebin. (Sql + Command)

① Sql> flashback Table <table name> To Before Drop;

② Ex: ⇒ flashback Table emp To Before Drop Rename to empl;

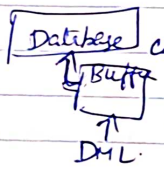
Purge:
Empty the recycle bin (ie) Emptying objects from the Recycle bin.
Select * from Recyclebin; (1 record
Sql> purge table tablename;
Select * from Recyclebin (0 record).

Comment // comment on table
comment on Table emp is 'Employee details';
Syntax: Comment on Table <Table name> is '<Text>';

Comment on Column
Sql> Comment on column <Table_name.col_name> is '<Text>';

DML Commands

⇒ deal with the data only.
⇒ interact with buffer and then to database.



⇒ Slower than DDL commands.
⇒ Can perform Roll back operations (since it is in buffer temporarily).

- Insert → Add/insert new data to the table.
- Select → retrieval of Info.
- Update → modify the data existing.
- Delete → remove the data
- Merge(9i) → Insert & update } together.
  (log) → Delete }

Insert

1). Insert into Emp (c1, c2, ...) values (v1, v2, ...);
⇒ for all the columns in the same order as we create. column name is not needed.
⇒ for columns, not in order → column name is not needed:
⇒ Left over columns will have Null value.

Substitution variables. (Multiple Insertion)
Insert into Emp values (&v1, '&v2', &v3);
→ try to the execute the before successful statement.

Insert into Emp1 select * from Emp;
↳ Inserting records using records of another table.

Insert into Emp1 (ENO, Ename, Sal) select (ENO, Ename, Esal) from emp;

Select

Selecting All Records with selected columns can select in our own order
1) Select ENO, Ename, from Emp; (selected order)
2) select * from emp;

Selecting particular/group of Records with all Columns.

Select * from Emp where Eno = 123;
Select * from emp where ENO = 123 or ename = 'aaa';

Selecting particular/ group of records with selected columns

1) Select Ename, Sal from emp where ENO = 123;
2) Select Sal + comm from emp where Sal = 2000;
3) select Sal * 2 from emp where Sal = 2000;

Update

1) Update emp set ename = 'aaa'; — update All Records
2) update emp set ename = 'aaa' where ENO = 1; // Particular Record
3) update emp set ename = 'aaa', Sal = 2000 where Ename = 'aa';
4) update emp set ename = 'aaa' Sal = 2000 where Ename = 'aa' And Sal = '1000';
5) update emp set ename = 'aaa' sal = 2000 where ename = 'aa' or ENO = 123.

Delete

1. Delete (from optional) emp; All records
2. Delete emp where ename = 'aaa'. // particular record.
3. Delete emp where ename = 'aaa' and ENO = 123;
4. Delete emp where ename = 'aaa' or ENO = 123;

Delete	Truncate
1) DML command	D.D.L command
2) Rollback is not possible	not possible. ✓
3) Implicit Commit not here	
4) All Rows of Table can be deleted	All rows of Table can be Truncated
5) Single Row of Table can be deleted	single row cannot be deleted.
6) where clause can be used	where clause cannot be used.
7) Multiple rows can be deleted	X
8. Trigger (upto 6) cannot be create	X
Trigger Can (upto	DR

performance

9. performance is slow because of buffers involvement.

performance is fast

10. Space removal/released is not here

space is released

Merge

⇒ introduced in 9i (Insert + update)

⇒ with a single statement we can perform three operations (insert, update, delete).

→ In 10G Insert + update ✓
update + Delete ✓
Delete + Insert ✗
Insert ✓
update ✓
Delete ✗
Insert + update + delete ✓

9i
Insert + upd ✓
Insert ✓
update ✗

⇒ Taking data from source table, operation performed in Target table
⇒ Target table should be / data from diff
⇒ ^ Source table can be multiple

merge into emp1
Set Linesize 200;

Diff b/w Database Manager & Database Admin

DBM
① manage and update a database

DBA
* They do not delete or create a database from scratch.

* DBA works to create or design, update, manage and delete databases.
* Admin has more freedom & responsibilities compared to DBA
* DBA is less of technical role and more of a business role

DBA

- Developing, Administering & maintaining Databases.
- Determining, the purpose and type of the database needed by a company.
- Understanding the requirements for data storage and accessibility.
- Designing the format and structure of objects present in the database.
- Assigning user rights

DBA → is a person or group in charge for implementing the database system within an organization. DBA has all the System privileges allowed by DBMS and can assign and remove access.

DBA also responsible for the evaluation, selection and implementation of DBMS package.

DBA

DBA → (person)

① Schema definition

The original database schema, is created by writing set of definitions which are translated by the DDL Compiler to the set of tables that are permanently stored in the data dictionary.

DDL → DDL compiler → Data definition → Dictionary

② Integrity Constraint Specification:

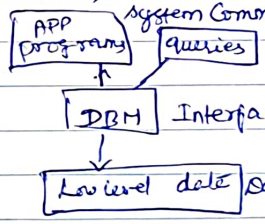
Integrity constraints are kept in a special system structure that is consulted by the DBM

DBM (program)

② Interaction with the file manager

DBM is responsible for the actual storing, retrieving & updating of data in the database.

DBM translates DML → Low-level file system Commands.



Integrity Enforcement

DBM determines whether updats to the database result in the violation of the Constraint. If so appropriate action must be taken.

Ex:: Acc balance should not go below the minimum allowed

③ Security - Granting of Authorization for data Access.

DBA regulates the access of the database, by granting diff types of authorization to the various users.

④ Storage structure and Access method definition

Appropriate storage structures and Access methods are created by set of definition which are translated by ~~data~~ compiler.
Data storage & definition Language compiler.

⑤ Schema and physical organization modification

modifications to either the database schema or the description of the physical storage organisation are accomplished by writing a set of definitions which are used by either DDL or DSL and DL Compiler to generate modifications to the appropriate internal system tables (data Dictionary).

Security Enforcement

It is the JOB of DBM to enforce the Security requirements.
EX:- Unauthorized users should not access the database / Authorized users should able to access the database.

Backup and Recovery

Like anyother electrical device, computer system is also subject to failure, information is ~~may~~ database may lost.
It is the Responsibility of the DBM to detect such failures & restore the database to a state prior to the failure.

Concurrency Control.

When several users update the database Concurrently. Consistency of data may no longer preserved.
Controlling the interaction among users is another Responsibility of DBM.

Database Users.

① Application programmers:- are Computer professionals, who write application programs. These programs could be written in programming language like C, VB, COBOL etc. Java, .NET, PHP.

* They interact with the system through DML calls which are included in a program written in host Language.
* Since the DML syntax and host Language syntax are diff

A special preprocessor – DML precompile → DML statements – normal procedure calls
Then it is run through the host lang compiler → appropriate Object Code.

Sophisticated Users.
⇒ Interacts with the system without writing pgms.
⇒ Instead they form their request in a query lang → query processor → DML statement → Instructions → DBM can understand

Specialized Users:-
⇒ Write specialized database Applications that do not fit into traditional.
⇒ CAD, systems that store data with Complex data types (Ex: graphics data & audio data)

Naive Users:-
⇒ unsophisticated Users interact with System by invoking one of the previously written permanent appln programs.

Overall System structure:-
⇒ A database system is divided into modules that deal with each of the responsibilities of the overall system.
⇒ The functional Components of database System are divided into

Query processor Components	Storage Manager Components
1. DML Compiler	1. Authorization & Integrity manager
2. Embedded DML precompiler	2. Transaction Manager.
3. DDL Interpreter	3. File Manager
4. Query Evaluation Engine.	4. Buffer Manager.

Physical System Implementation
1. Data files
2. Data Dictionary.
3. Indices
4. Statistical Data

DML Compiler → DML statements in QL → Low Level Instruction that Query Evaluation Engine understands

2. Embedded DML precompiler → DML statements embedded in an appln prog to normal procedure calls.

* precompiler interacts with query processor to generate appropriate code

3. DDL Interpreter: interprets DDL statements & records them in a set of tables metadata

4. Query Evaluation engine → executes low level instructions generated by DML compiler

5. Authorization & Integrity manager → tests for the satisfaction of Integrity constraints and checks the authority of users to access data.

6. File manager: manages the allocation of space on disk storage & data structures.

7. Transaction manager - which ensures database consistency and concurrent transaction

8. Buffer manager: responsible for fetching data from disk storage into main memory.
→ decides what data to cache in memory.

9. Data files → which store the database itself

10. Data dictionary - which stores the metadata about the structure of the database.

Indices → provide fast access to data items holding particular values.

Statistical data: which store statistical info about the data in the database.

Ex:2 Aggregate functions

⇒ Aggregate function → performs a calculation on multiple values and returns a single value.

⇒ used to perform the calculations on multiple rows of a single column of a table. It returns a single value.

* Count() → Counts how many rows are in a particular column (where)
* max() → function is used to get the max value from a column.
* min() → function is used to get the lowest value in a particular column.
* Avg() → calculates the average value of a group of selected values.
* distinct() → used to select the distinct rows.
* Sum() → used to get sum of all the values in a particular column.

Unit-II

2. Entity Relationship model.

⇒ ER model describes the structure of the database with the help of a diagram which is known as Entity Relationship diagram (ER diagram).

⇒ ER model is a blueprint of a database, that can be later implemented as Database.

⇒ ER data model is a collection of basic objects called Entities and the relationship among those objects

Entity & Entity sets

⇒ Entity is an object and is distinguishable from other objects. It can be identified uniquely

⇒ In terms of RDBMS entity is a table or attribute of a table in database.

Ex: John with Aadhar no is an entity uniquely identified in the world.

Tangible Entity - Entity that exists in the real world physically.

Ex: person, car, house.

Intangible Entity - Entity that exist only logically and have no physical Existence.

Ex: Bank, Account.

Entity set:-

→ is a collection of entities of some type.

E1
E2
E3 Entity set

① Ex: The collection of all the students from the student table is an example of entity set.

Ex2 The collection of all the employees from the employee table

Table name: Student

studentID	name	Age
1		
2		
3		

→ entity.

Attributes - characteristics of the entity.

⇒ Attribute → describes the property of an entity.

⇒ represented as ovals in ER diagram.

⇒ An entity may contain any no. of attributes

Domain:- is a set of permitted values for an attribute in a table.

Ex:- Domain of customer_name (attribute) → might be all text strings of certain length

(2) domain of attribute account no → might set of all positive integers.

(3) telephone no must be to positive integers.

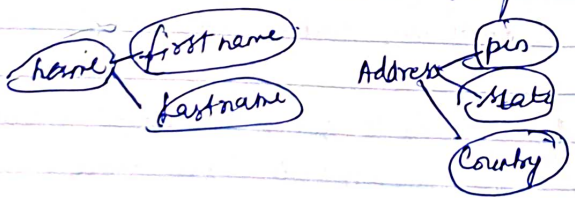
Every Entity is described by a set of (attribute, data values) pairs. There is one pair for each attribute of the entity set.

Ex:- a particular student entity is described by a set {(name, Lee), (Aadhar 3---), (Street, ...), (city,)}.

Types of Attributes

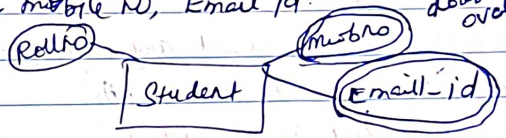
(1) Simple Attributes / Atomic Attributes
Attributes that cannot be classified further.
Ex:- Roll no, Account no, Age

(2) Composite Attribute:-
Attribute can be divided into subparts
Ex:-

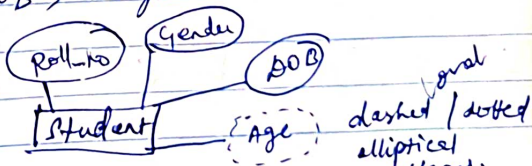


3) Single valued
The attribute holds single value is called single valued.
Ex: Roll-no, Age, DoB, Gender.

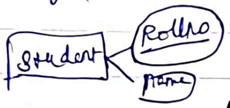
4) multivalued
Attribute can take more than one value.
Ex. mobile no, Email id. double ovals.



5) Derived
=> Those attributes whose values can be derived from the values of other attributes.
=> They are always dependent upon other attribute for their values.
Ex:- From DOB attribute, we can derive the Age attribute
Here Age → derived Attribute
DoB → single-valued attribute.



6) Key Attribute → special attribute that uniquely identifies the entity in the entity set. The value of Key attribute must be unique.

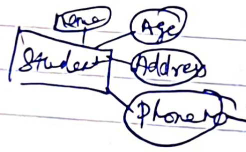


EX:- Emp id → key attribute for employee Entity.

7) Null valued Attribute:-

An attribute, which has not any value for an entity is known as Null valued Attri

EX:- Assume student is an entity.



→ no phone → null valued attribute

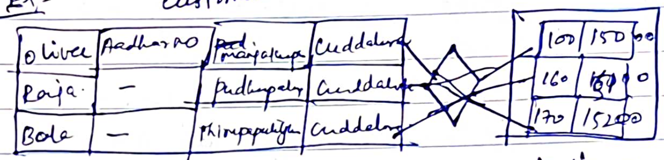
Relationship

⇒ It is an association among several Entities.

⇒ represented by diamond shape in ER diagram.

EX

EX:-



Entity 1 Entity 2.

we may define a relationship which associates Raja with acc no 100

Raja is a customer with acc no 100

Relationship set → set of relationship of the same type

binary Relationship set (degree of Relationship 2)

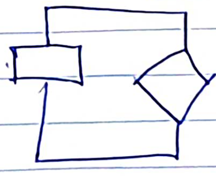
↓ If two entities are involved in a relationship →

Ternary Relationship set → If three entities are involved in a relationship

Role The function that an entity plays in a relationship is called Role

Unary Relationship

degree of Relationship -1



no. of entities that participate in a relationship

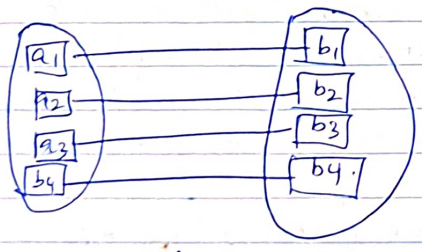
Mapping constraints / Cardinalities:

It tells the number of entities to which another entity can be associated through a 'relationship set'.

⇒ Mapping cardinalities is most useful in describing binary relationship set

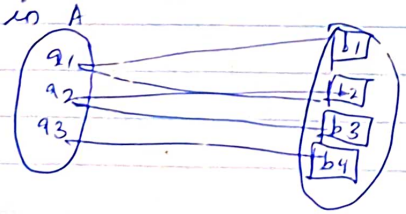
One to one (1:1)

An entity in A is associated with atmost one entity in B and and an entity in B is associated with atmost one entity in A



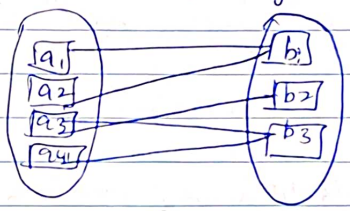
One to many (1:n)

An entity in A is associated with any number of entities in B. An entity in B however, can be associated with atmost one entity in A



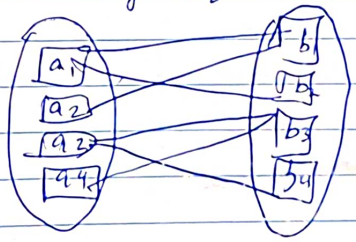
Many to one (n:1)

An entity in A is associated with atmost one entity in B. An entity in B, however, can be associated with any number of entities in A



Many to many (n:n)

An entity in A is associated with any number of entities in B and an entity in B associated with any no of entities in A



E-R diagram

The overall logical structure of the database can be expressed graphically by E-R diagram.

→ ⇒ directed line } To distinguish
— - undirected line } we draw a
directed line or an undirected line between the relationship set.

Keys ⇒ ensure no duplication of information
→ plays an important role in relational database.
⇒ uniquely identifies any row or record of data from the table.
⇒ also used to establish and identify relationships b/w tables.

Primary Key
⇒ PK is used to ensure data in the specific column is unique.
⇒ only one primary key is allowed in a table.
⇒ It does not allow null values.
⇒ Its value cannot be deleted from the parent table.
⇒ It is a candidate key

Candidate Key → subset of super key
⇒ values should be unique and non-null
⇒ every table has to have at least one Candidate key.
⇒ All CKs are super keys but not vice versa.
⇒ Many Candidate keys are allowed.

Super key ⇒ is a single key / group of multiple keys. ⇒ can contain multiple attributes that might not be able to independently identify tuples in a table; but when grouped with certain keys they can identify tuples uniquely.
Ex:- (Id, Name) (Id, email) (Id, Name, email)
⇒ can contain redundant attributes.
⇒ many super keys are allowed

Foreign key ⇒ The table with the foreign key → child table
The table with the primary key → parent table
⇒ is a column or group of columns in relational database table that provides a link b/w data in two tables.
⇒ it is a column that references a column of another table
⇒ It refers to the field in a table which

is the primary key of another table.
=> more than one foreign key are allowed in a table.
=> It can contain NULL values.
=> It can contain duplicate values.

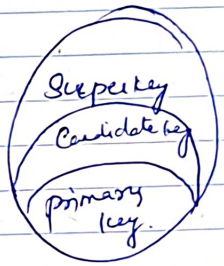
Student.

Rno	name	e-mail	marks
1	Ramesh	@gmail	95
2	Sathish		97
3	mahesh		97
4	Mahesh		94

keys -> Attributes or collection of attributes, which can be used for unique identification of each row in a table.
=> can make R.no as key but not name
=> can make email as key, not marks.

1. Super key
Same of the key.
Ex: 'Rno, email
(Rno + name)
(Rno + email)
(email + marks)
(Rno + name + e-mail)
(Rno + name + email + marks)
(name + marks) X
↓ not key.

2. Candidate key -> minimal subset of super key that can be used for unique identification of each table.
Rno ✓, e-mail,
e-mail
roll + name X
email + marks X
Rno + name + email X
Rno ✓



3. primary key.

Rno

Referential Integrity.

4. Foreign keys.

primary key.
parent table.

Persons table

personID	Last name	First name	Age
1	Kumar	Raj	30
2	Gupta		23
3			24

Orders Table — child table.

Order ID	Order Number	person ID (foreign key)
1	77895	3
2	44678	3
3	22456	2
4	24562	1

(person ID: these have to be one of the values in the parent table)

person ID in orders table points to the person ID in "person" table.

Table : orders

Order ID	product	Total	customer ID
1	paper	500	5
2	pen	10	2
3	marker	120	3
4	Books	150	4
	pen	50	5

Table : customers

customer id	First-name	last-name	age	country
1	John	Mathew	31	UK
2	Rajesh	Kumar	22	India
3	Kumar	Gupta	25	USA
4	Bala	murugan	22	India
5	mahesh	Raj	30	UK

Extended ER features / Enhanced ER (EER diagram) developed

1. Generalization
2. Specialization
3. Aggregation.

In order to handle the complex application better (Improved features of ER diagram)

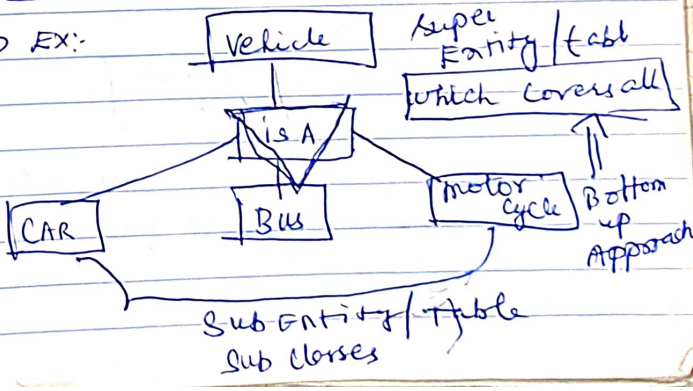
1. **Generalization** → states that something in common.

⇒ is the process of extracting common properties from a set of entities and create a generalized entity from it.

⇒ Generalization is a "Bottom up approach" in which two or more lower level entities can be combined to form a higher level entity. if they have same attributes in common.

⇒ "sub entities" are combined to make a "super entity"

⇒ EX:

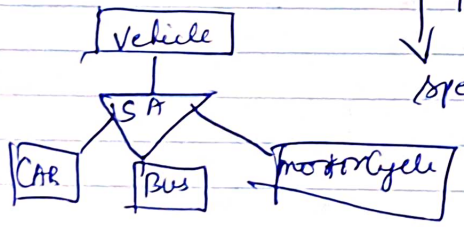


2. Specialization

=> Specialization is opposite to Generalization
=> In specialization, an entity is broken down into sub entities based on their similar characteristics. (lower level entities)
=> Specialization is a Top down approach where higher level entity is specialized into two or more lower level entities.

Ex:-

Super class is defined first, the subclass and its related attributes are defined next and relationship set are then added.

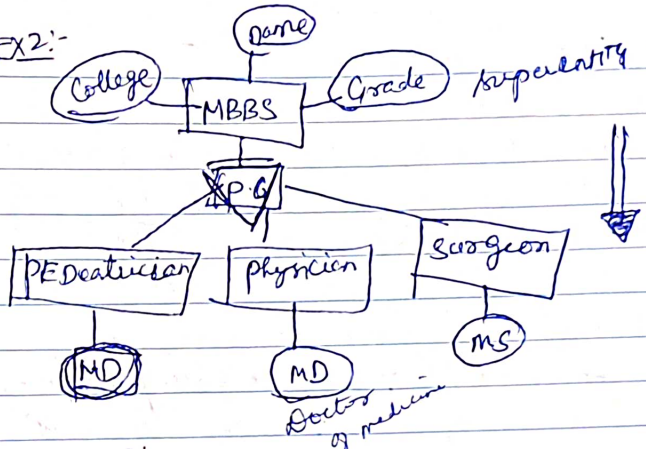


Top down approach

Specialization

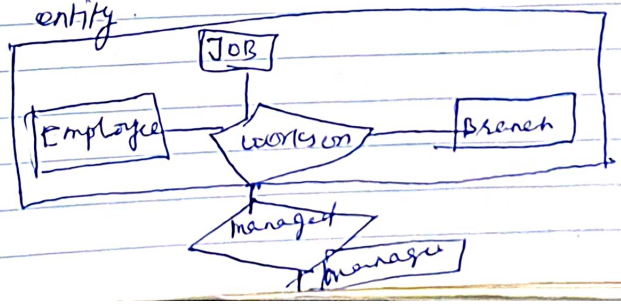
For applying generalization & specialization the structure of resultant figs are same.

Ex2:-



Aggregation

=> (2) Aggregation is used when we need to express a relation among relationship
=> Aggregation is an abstraction through which relationships are treated as high level entities
=> Aggregation is a process when a relationship b/w two entities is considered as a single entity and again this single entity has a relationship with another entity.

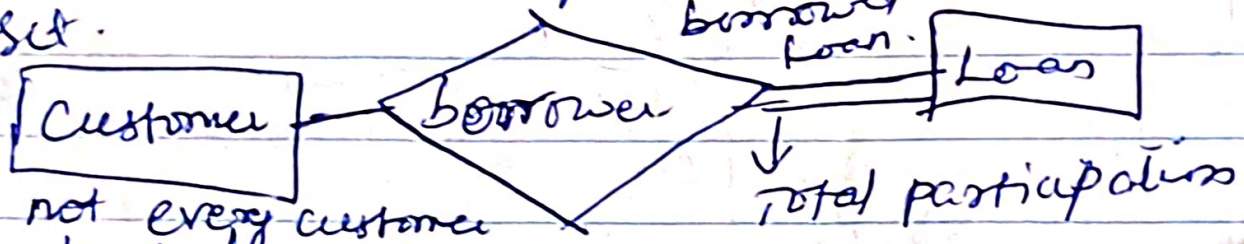


NCSIT 301

Participation Constraints

Total participation.

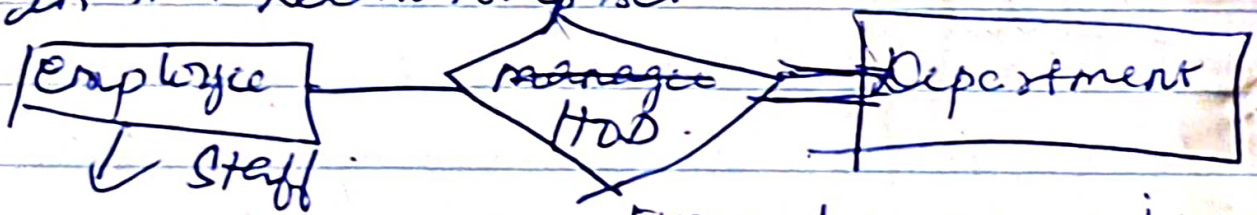
Every entity in the Entity set have at least one relationship in that relationship set.



not every customer is a borrower.

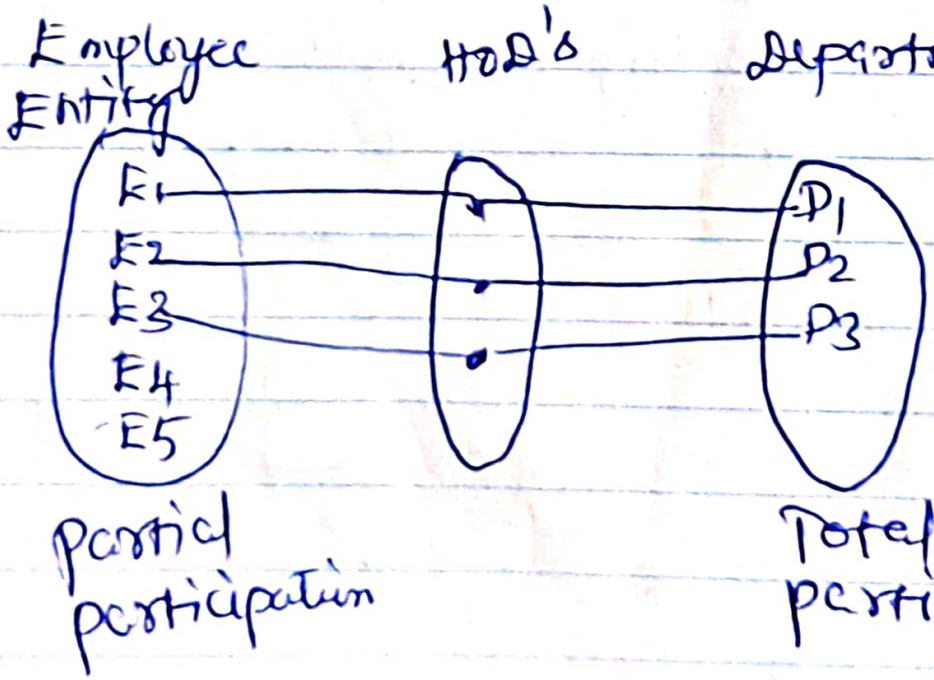
partial participation

Each entity in the entity set may not occur in at least one relationship in that relationship set.



not every employee manages a department

Every department is managed by at least one employee called manager



Gate vidyalay

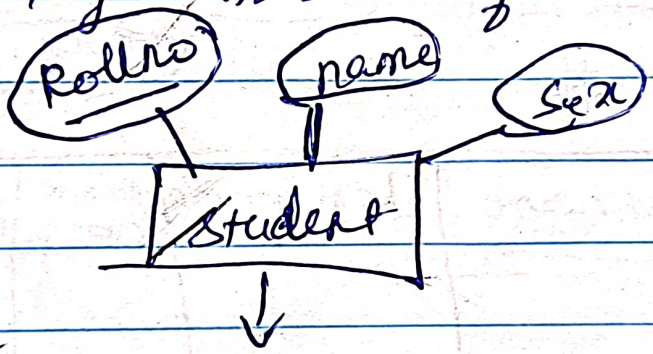
Converting ER Diagram to Tables:-

Rule 01:- For strong Entity set with only simple Attributes.

A strong Entity set with only simple attributes will require only one table in Relational model.

- Attributes of the table will be the attributes of the Entity set.
- The primary key of the table will be the key attribute of the entity set

Ex:-



Student

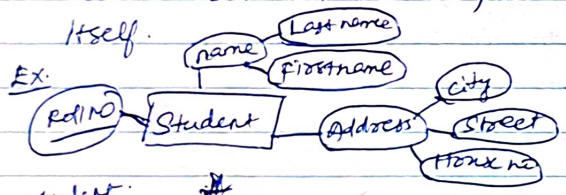
Roll no	name	sex

Schema: Student (Roll no, name, sex)

Rule 02: For Strong Entity sets with Composite Attributes:-

- A strong Entity set with any number of Composite attributes will require only one table in relational model.
- while conversion, simple attributes of the Composite attributes are taken into account and not the composite attribute itself.

Ex:



Student

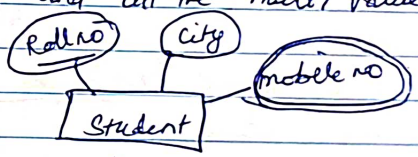
Roll NO	First name	Last_name	House no	Street	city

Schema : Student (Roll NO, First name, Last name, House no, street, city)

Rule 03: For strong Entity set with multi-valued Attributes.

- A strong Entity set with any number of multivalued attributes will require two tables in Relational model.
- One table will Contain all the simple attributes with the primary key.
- other table will Contain the primary key and all the multi valued attributes.

Ex:-



Roll no	city

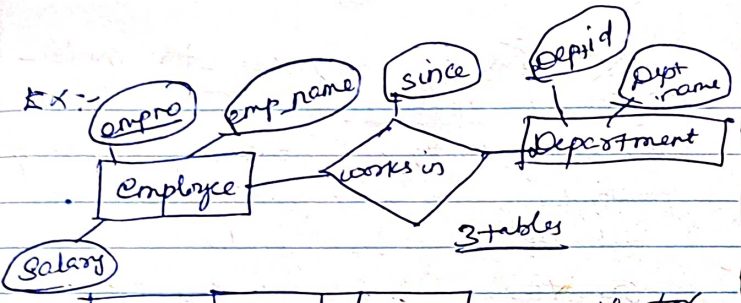
Roll no	mobile no

Rule 04: Translating Relationship set into a Table:-

=> A relationship set will require one table in the relational model.

- Attributes of the table are:
  - primary key attributes of the participating entity sets
  - Its own descriptive attributes if any.

Ex:-



3 tables

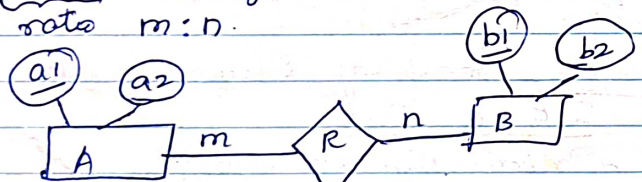
Empno	deptid	since

- one table for entity set 'employee'
- one for 'Department'
- one for relationship set "works in".

Rule 05:- For Binary Relationships with Cardinality Ratios:-

The following four cases are possible.

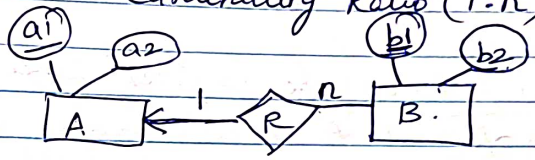
Case-01:- Binary relationship with Cardinality ratio m:n.



Here three tables will be required:-

1. A(a1, a2)
2. R(a1, b1)
3. B(b1, b2)

Case 02: For Binary Relationship with Cardinality Ratio (1:n) one to many



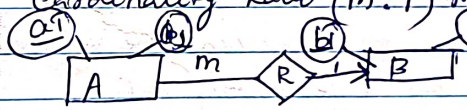
Here two tables will be required

1. A(a1, a2)
2. BR(a1, b1, b2) fk

The Combined table will be drawn for the entity set B and Relationship set R.

Case 03:

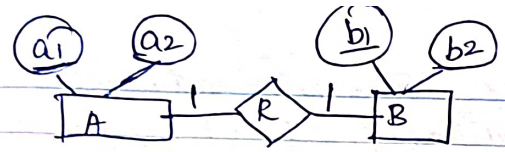
For Binary Relationship with Cardinality Ratio (m:1) many to one.



Here two tables will be required.

1. AR(a1, a2, b1) fk.
2. B(b1, b2) → The combined table will be drawn for A and R.

Case 04: For Binary Relationship with Cardinality Ratio 1:1



Here two tables will be required. Either Combine R with 'A' or 'B'

way 1

1. AR(a1, a2, b1) (fk)
2. B(b1, b2)

way 2

1. A(a1, a2)
2. BR(a1, b1, b2) (fk)

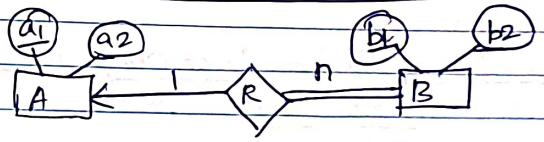
Note: for m:1 or 1:m always the many side - a combined table will be drawn for many side entity and the Relationship set.

=) For 1:1 you can combine the relationship with any of the entity sets.

Rule 06:- For Binary Relationship with Both Cardinality Constraints and participation Constraints

- Cardinality constraints will be implemented as discussed in Rule-05
- Because of the total participation Constraint, foreign key acquires Not NULL Constraint (ie) now foreign key can not be null

Case-01: For Binary Relationship with Cardinality Constraint and Total participation Constraint from one side



Because the cardinality ratio = 1:n, we will combine the entity set B and the Relationship set R.

Then two tables will be Required

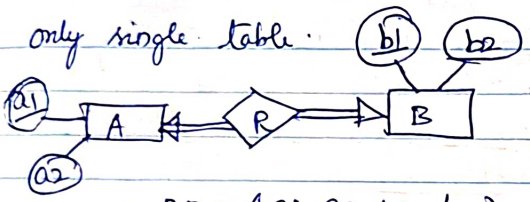
1. A(a1, a2)
2. BR(a1 (fk), b1, b2).

Because of the T.P foreign key, a_1 has acquired not null Constraint.

Case-02 For BR with CC and TPC from BOTH sides.

- If there is a key Constraint from both the sides of an entity set with Total participation, then the binary Relationship is represented by using

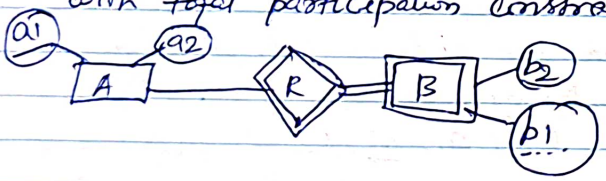
only single table.



- ARB (a1, a2, b1, b2)

Rule 07: For Binary Relationship with weak Entity set.

weak Entity set always appears in association with identifying relationship with total participation constraint.



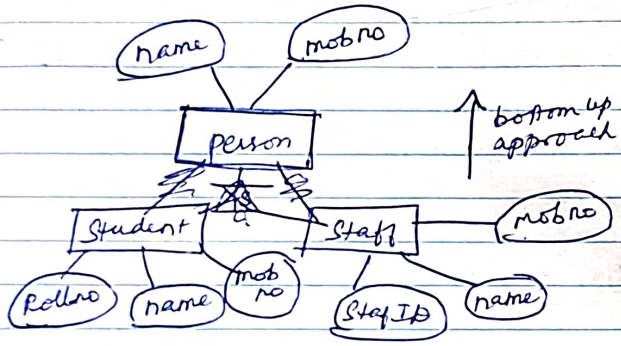
Here two tables are required

1. A (a1, a2)
2. BR (a1, b1, b2)

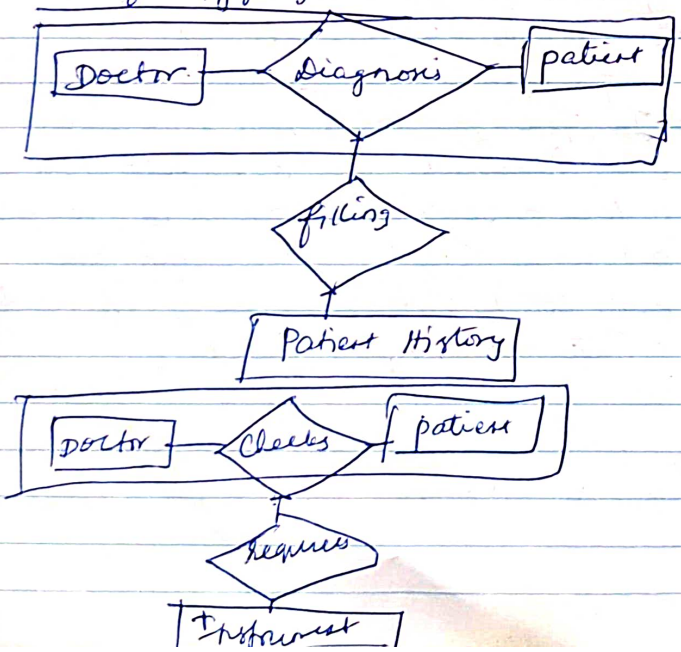
Diff b/w Generalization & specialization

Generalization	Specialization
1. Generalization works in Bottom up approach	It works in top-down approach.
2. Size of schema gets reduced.	Size of schema gets increased.
3. Generalization is normally applied to group of entities.	we can apply specialization to a single entity.
4. In Gene what happens is it takes the union of two or more low-level entity sets to produce a higher level entity sets.	Speci - is reverse of Generalization. It is the process of taking a subset of higher level set to form a lower-level entity set.
5. There is no inheritance in generalization	There is inheritance in specialization

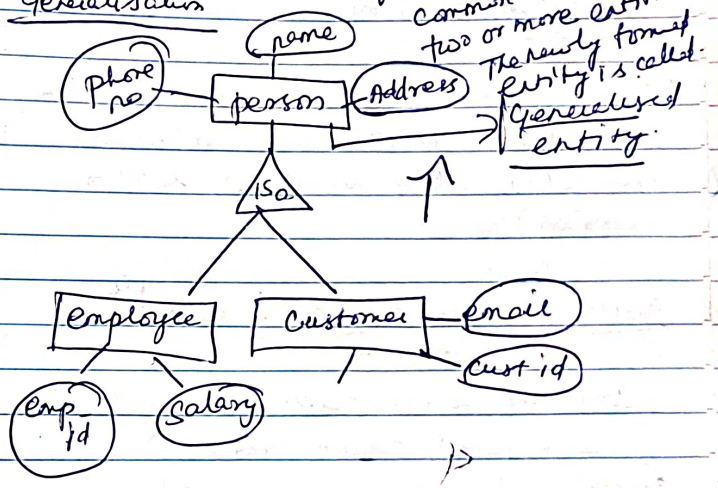
Ex 2 Generalisation



EX 2 Aggregation



Generalisation is a process in which a new entity is formed using the common attributes of two or more entities. The newly formed entity is called Generalised entity.



partial Generalization: Some higher-level entities may not belong to any lower-level entity set.

Total Generalization: The higher level entities completely belong to all lower level entities.

primary key at Table

1) Create table employee(id number(5) primary key, name char(20), dept char(10), age number(2), salary number(10), location char(10));

Syntax to define a primary key at ~~table~~ table level.
optional.

[Constraint constraint_name] primary key((Column_name1, Column_name2..)

Syntax to define a primary by at column level

Column name datatype [CONSTRAINT Constraint_name] primary key.

NOT NULL Constraint

1) Create table employee (id number(5), name char(20) constraint nn_nn NOT NULL, dept char(10), age number(2), salary number(10), location char(10));

Select

select firstname || ' ' || lastname from employee;

Alias

select firstname || ' ' || Lastname AS empname from employee;

select firstname As name from studentdetails;

Insert

Insert into tablename (col1, col2, col3..) values (value1, value2, value3 ... values);

order by

select * from Emp orderby ename;
//Ascending is default

select * from Emp orderby ename desc;

12. Select the student failed in subject 1
=> Select * from Student where Sub1 < 40;

13. Select the students who got above 60 and less than 90 in sub2.

=> Select * from student where ((Sub2 > 60 and sub2 < 90));

14. Select the female student who secured total above 350
Select * from student where ((Sex='F') and (total > 350));

15. Update age 20 for all the students whose ages greater than 20.
=> update student set age = 20 where age > 20;

16. Select the male student who got total less than 200
=> select * from student where ((Sex='M') and (total < 200));

17. Delete all the failed female student
=> delete from student where ((Sex='F') and (result='fail'));

18. Delete all the students who got total less than 250.
=> delete from student where total < 250;

19. Modify the name field length to 25.
=> Alter table student modify name varchar2(25);

20. Add a new column rank number of 3
=> Alter table student add rank number(3);

Special operators:-
1) like
2) In
3) between and AND
4) Is null.

Like Operator:- is used to select the field values starting or ending with some alphabets.

Select the student whose name ends in letter a.

21) Select * from student where name like '%a';
Select * from student where name like 'a%';

22) Select * from student where age in (18, 20, 22);
In op → used to select the multiple values.

23) Between → used to select the range of starting to ending values.
Syntax: Select * from student where Sub1 between 60 and 90;
300 400

23) Null :- Select * from tablename where Columnname is null;
Ex. Select * from student where age is null;

24. Sort :- Arrange the records by name
Select * from student order by name;
Select * from student order by desc;

25 Groupby :- The groupby clause groups rows - If you want to group the number of customers in each city.
Syntax :-
Select (Columnname) from tablename where Condition groupby (Columnname) orderby (Columnname);

1. To list the number of customers in each city.
→ select count (Customer-id), city from customers group by city;

→ Select the student based on the department
Select count (regno), dept from Student group by dept;

DCL
Create User shastri identified by mary;
User create
Alter user jus identified by sam;

Grant
Grant Connect, resource to shastri;
Grant succeeded.

Grant select, update on emp to jus;
Grant succeeded.

Revoke:-

Revoke connect, resource from jus;
Revoke succeeded;

1) Arithmetic operators *, +, /, -
2) Comparison operator =, <=, >=, !=
3) other operators a) between and
   b) 'S%'
   c) is
4) Logical operators: a) salary > 10000 and department no = 10.
   b) salary > 10000 or department no = 10.
   c) not between 11000 and 15000.
   d) not in (11,000 and 15000);

Insert into student values ('&rollno', '&name', '&sex', '&age', '&dept', '&sub1', '&sub2', '&sub3', '&sub4', '&total', '&avg', '&result', '&doj');

Result:

update student set result='fail' where ((sub1<=40) or (sub2<=40) or (sub3<=40) and (sub4<=40));
or.
update student set result='pass' where ((sub1>=40) and (sub2>=40) and (sub3>=40) and (sub4>=40));

Nested queries:- Query within a query.

Second highest value.

select max(fieldname) from tablename where fieldname < (select max(fieldname) from tablename);

eg:- select min(max(sub1)) from CSC2023 where sub1 < (select max(sub1) from CSC2023);

- Select sub1 marks from 40-80 also select sub1 marks from 70-80.

Eg:

Select * from student where ((sub1 >=40) and (sub1 <=80) union select * from student where ((sub1 >=70) and (sub1 <=90));

union all

Select columnname from tablename where condition union all. Select columnname from tablename where condition;

Eg

Select * from student where ((sub1 >=40) and (sub1 <=80)) union all select * from student where ((sub1 >=70) and (sub1 <=90));

output:-

Produce the remaining result of first query which are not present in the second query.

- update student set result = case when ((sub1 >40) and (sub2 >40) and (sub3 >40)) then 'pass' when ((sub1 <40) or (sub2 <40) or (sub3 <40)) then 'fail' end;

Sql Select Statement - Sql command is SQL SELECT

The most commonly used statement.

* The Sql SELECT Statement is used to query or retrieve data from a table in the database.
* A query may retrieve information from specified columns or from all of the columns in the table
* To create a simple SQL SELECT statement you must specify the column(s) name and the table name

Ex:- mandatory mandatory.

Select empname, salary from employee where empid = 100;

optional.

⇒ If we want to display the first and last name of an employee combined together, the Sql statement would be like.

select firstname || ' ' || lastname from employee;

O/P firstname || || lastname

Rahul Sharma.

priya chandra.

You can provide aliases as below.

Select firstname || ' ' || lastname AS emp_name FROM employee.

o/p emp_name.
Rahul Sharma.
Arjun Devi.

⇒ Expressions can also be used in the where clause of the SELECT statement.

If we want to display empname, current salary and @ salary * 1.2 as new salary.

Select name, current salary, salary * 1.2 AS new_salary from employee where emp id = 005;

o/p

name	current salary	new salary
priya	30000	36000.

1. Explain Dql statements with examples?
2. Explain DML statements with example?
3. Explain SQL aggregate functions with example
3. Explain set operations

August (2019) 4X5=20

1. Explain primary keys with Example
2. Define database & List out the purpose of Database.
3. What is Data Independence. Discuss about two levels of DI
4. Define Entity & Entity sets.
5. Explain the ER diagram with a neat example.
6. List functions of DBA.

Part-B 3X10=30.

7. Overall System structure.
8. Discuss the mapping Constraints
9. Define the Record based logical model with example.
10. Create a student excel sheet table with minimum 5 fields. Insert records and add a new field name.

(2) Part-A

1) Write short notes on Database Languages.
2. Exp the types of Attributes.
3. Discuss about the level of Data abstraction
4. what is DBMS? Explain
5. Types of Database Users? Explain
6. Explain the three types of Alter Commands.

Part-B

1) Disadvantages of file processing System? Explain
2) Overall database system with neat diagram
3) mapping cardinalities with examples?
4. Explain the Symbols used in the E-R diagram with examples
5) write the answers for the following queries using the below student table.

Regno	name	Age	Sex	mark1	mark2	Total	result

(i) Select the male student who failed in both the subjects.
(ii) Add a new column Average number(5,2)
(iii) update the student age: 20 for the Regno=102
(iv) Delete the student whose Reg NO=103
(v) Select the male student who got cent. in marks

Regno	Name	Age	Sex	mark1	mark2	Tot	R
101	Arun	20	male	20	25	45	Fail
102	Suresh	18	male	60	80	140	pass.
103	malar	19	F	30	20	50	F
104	Sojia	20	F	100	90	190	P.
105	Rani	18	F	20	80	100	F
106	Ramesh	21	M	90.	100	190	P

Part-A

2. write short notes on Distances & Schemas.
3. what is Data Abstraction?
4. Define DDL, DML, and DCL with Examples.

Part-B

a) Create a table with minimum 4 fields
b) Add a new column to the table
c) Insert two records to the table
d) Delete one record from the table.

Views

To create view => first we have to get privilege / permission from DBA. For that

```
SQL> Connect system;
Enter password: manager;
  Connected.
SQL> Create view [view-name) as select
Column list A from table-name [where condition];
```

=> select statement is used to define the columns and rows that you want to display in the view.

View

Virtual table.

A view is also like a table, but does not maintain physical data,

① View is a database object. It contains logical copy of the data from table. View is created on the table,

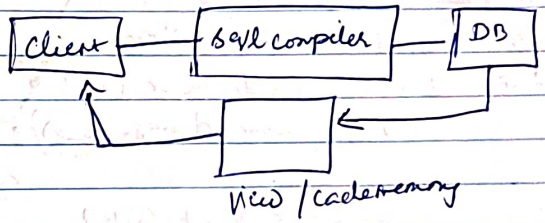
=> why we need a logical copy or a view?

=> View provides security.

=> Increases / improves DB performance

Query Execution process

Ex: Suppose we want to execute select query. when u submit the query. Oracle Database.



If you're accessing the database increases performance decreases.

=> If the same user / diff user access the same information repeatedly / frequently executing the same query / multiple times. Ex: Availability of tickets.

=> Don't get the data from the table, Instead Create a view that is taking the logical copy of that data and place it in view. From then on, the data will take from the view. First time only it get the data from the database. From second time onwards the data will be get from view.

Sql> grant all on employee to shakira;
grant succeeded.

Create Table tablename (fieldname datatype (size) Primary key, fieldname datatype(size) not null, fieldname datatype(size) unique, fieldname datatype(size). check (fieldname in (value1, value2, ... valuen) fieldname datatype(size));

Ex:

Create table Stud(Rollno varchar2(10), primary key, name varchar(30) not null, DNO number(5) unique, age number(3) not null, sex char(1) check(sex in ('M','F')), total number(3) not null);

Select user from Dual; // to know which user we are login

Select * from all_users; To see who are all the users of database.

TO create user identified

Create user username identified by password;
Grant resource, connect to username;

User name: shakira
password: joe.
particular user.
Grant select on employee to mary;
Grant succeeded

User name: mary
password: shakira.

Select * from shakira.employee;
(must)
Table will be displayed. without shakira, table view does not exist.

The change in table will reflect only after giving Commit;

Learn Code today & Code.

Revoke

getting back the previleges:

Revoke select ON tablename from username;
Revoke ALL ON " " " ;

public:-

TO Grant Resource to all the users;

Syntax:

Grant select on <table_name> TO public;
all users in the database can access the table.

Revoke " " " " from public

Role :- like watsapp group.

Create role <roll name>

Grant <role name> TO <user1> <user2>

Ex.

Create roll T1;

grant T1 to mary, ericka;

grant select on employee to T1;

grant succeed.

Now go to mary. To make the effect logout and login again.

Revoke <role name> From <user1>.

Primary key - Unique Constraint (violated). cannot be null.

not null

Cannot insert null value into ("Usernames" "Roles" "none")

~~Uniquekey~~

Unique Constraint violated.

Check key

Check constraint violated.

Alter into Tablename add Columnname='' into where

Oracle - kaashivinfotech.com/oracle-training-in-chennai

founded by Larry Ellison - Bob miner

Oracle Corporation Compatible with Internet

8i-9i → i → internet based database.

10g → grid based Architecture (can share hardware)

12C → Cloud Based Database.

TCL → A set of DML operations with a commit or roll back is called Transaction

Commit: Insert into stud. values ('101');
commit;

Rollback:

Insert into stud values ('102');
Insert into stud values ('103');
Insert into stud values ('104');
commit;

clear the temporary memory to cancel the transaction happened recently.

Insert into stud values ('105');
Insert into stud values ('106');
Roll back;

Savepoint → To make the points to make the level.

⇒ Rollback upto this level

insert into stud values ('101');
Insert into " ('102');
. " ('103');
Savepoint x;
Insert into stud values ('104');
" ('105');
" ('106');
Savepoint y;
Insert into . ('107')
Rollback to x; //Rollback complete.

Commit:
* Used to end your current transaction
* makes the changes permenant in Database.

Rollback
* Rolls back the current transaction, canceling its changes.

Savepoint:
* It is a point in a transaction in which you can roll the transaction back to a certain point without rolling back the entire transaction.

Example: First insert Extra two records to
⇒ a table and select and see the newly inserted records will be visible, but now it is in temporary memory.
⇒ If you give the rollback, the newly inserted records will be deleted, but the old records will be existed.
⇒ Again insert two new records & give commit;
⇒ Then give rollback the newly records will not get roll back, because they are permenantly stored in the database.

100
101
102
Commit;
103
104
rollback;

permanent database: 100, 101, 102
temporary mem: 103, 104 (X, X)

rollback complete;

101
Savepoint x;
102
103
104
Savepoint y;
105
rollback to savepoint x
EOT

Savepoint x;
Save point created;

group by clause → is used when we want to group with the column suitable clause is used to group the rows that have the same values.

* Group by is used with aggregate function
* [where clause] is used to filter rows.

Having clause: → used to

* It is used to restrict/filter the result returned by the Group by clause
* Having clause acts same as where clause.

Syntax: Select column name 1, column name 2.. from table name Group by column name1, column name 2 [Having condition];

Ex: ① Select dept, count(name) from student group by dept;

Dept	Count(name)
maths	3
CS	5
physics	2

② Select result, count(rollno) from student group by result;

Result	Count(rollno)
pass	6

select result, count(*) from emp group by dept Having Count(*) >= 3 Order by dept;

empid	ename	designation	salary
1	Ram	Analyst	50,000
2	Raja	programmer	60,000
3	Aadhi	HR	80,000
4	Raja	Sales	40,000
5	Mohan	HR	70,000
6	Suresh	programmer	50,000

Average salary.

To get job designation and average salary. Count(*) or count(empid).

② Select designation, avg(salary) from employee group by designation.

designation	avg(salary)	count(*)
Analyst	50,000	2
programmer	55,000	2
HR	75,000	1
Sales	40,000	1

Select designation, count(emp_id) from employee group by designation having count(emp_id) >= 1 Order by job_desc;

select designation, count(emp-id)
from employee where salary > 40,000
group by designation
having count (emp_id) > 1.
order by designation;

Unit - III

Ex: Relational calculus
query language
procedural / non procedural
what is Relational Algebra?
Relational Algebra is a procedural query language. It gives a step by step process to obtain the result of the query. It uses operators to perform queries. It consists of set of operations
Basic fundamental / operations. { what to do / how to do
1. select (σ sigma)
. It is used to find the tuples in a.

* Relational Algebra is a collection of operations on Relations / tables.
* Each operation takes one or more relation as its operands (input) and produce another (new) relation as its result (output).

Operations On Relational Algebra

Unary operation: operate on one relation
Ex: select, project

Binary operation: operation performed on pair of relation.
Ex: Union, Intersect
∪ ∩

Select Operation: It select tuples (rows) from a relation (table)
⇒ It is denoted by σ (sigma)
Syntax:

$$\sigma_{(\text{select condition})}(R)$$

↓ filtering Criteria.

Ex: $\sigma_{sid=1}(\text{student})$

sid	sname	dept	result
1	Aadhi		

$\sigma_{DNO=2}$ and salary > 40,000.

⇒ In selection operation all relational operators may be used $=, \neq, <, >, \leq, \geq$

Ex2: Display all details where id > 1.

$\sigma_{sid>1}(student)$

⇒ conditions may be combined using and ($\wedge$), or ($\vee$)

$\sigma_{sid=1 \text{ or } sid=3}$ → Either one condition is true.

$\sigma_{sid=1 \text{ and } sid=3}$ = both the conditions is true.

Project operation

It yields column (attributes) of a relation.

⇒ Using project operation, duplicate values are automatically removed

⇒ It is denoted by 'Π' symbol.

Syntax: $\Pi_{A1, A2 \ldots An}(R)$

$A_1, A_2, \ldots A_n$ → Attributes of a relation

R – Relation

Ex: $\Pi_{name}(student)$

output:

name
A
B
C
B.

A
B
C

⇒ duplicate values are avoided.

⇒ project & select operation can be used simultaneously

$\Pi_{name}[\sigma_{class=12}(student)]$

⇒ display the name of students whose class is 12.

name
B

rollno	name	class.
1	A	11
2	B	12
3	C	11
4	D	12

Cartesian product operation (or) Cross product ($\times$)

It combines R_1 and R_2 without any condition. It yields new relation which has a degree (no. of attributes) equal to the sum of the degrees of two relations operated upon. no. of columns/attributes = no. of columns of R_1 + no. of columns of R_2

No of rows = no. of tuples in R_1 × no. of tuples in R_2.

② The no. of tuples is product of no of tuples of two relations.

⇒ It is denoted by 'X' symbol.

Ex:-

Tables.

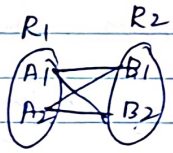
Total no of Attributes	R_1	R_2
	x	y
Tuples	n_1	n_2

$R_1 \times R_2$

$x+y$ → no of attributes

$n_1 \times n_2$ → no of tuples.

syntax:



A_1B_1 A_1B_2
A_2B_1 A_2B_2

R

A	B
α	2
β	1

S

C	D
2	101
y	102

R×S →

R+S = 2+2 = 4

A	B	C	D
α	2	2	101
α	2	y	102
β	1	2	101
β	1	y	102

Cust-Id	Cust name	Cust city

Union operation (∪)

It is used to select all the rows from two tables. Avoids duplicate rows.

Syntax ⇒ tables should have same no. of attributes

table name 1 ∪ table name 2.

Π Student name (Course) ∪ Π Student name (Student)

Intersection operation

It is used to select common rows from two tables, tuples present in both the relations.

syntax:-

table name 1 ∩ table name 2

Π student name (Course) ∩ Π Student name (Student)

Set difference (−)

If we want to select all those tuples that are present in Relation R_1 but not present in relation R_2 can be done using set difference $R_1 - R_2$

table name 1 − table name 2

Π studentname (Student) − Π studentname (course)

Course

Rename (ρ) ← changing name of the relation
changing name of the attribute
changing both
fetching the column names and renaming the resulted relation.

→ can be used to rename a relation

ρ_{newname} (Relation name)

ρ (cust_names, Π (customer_name) (Customer))

↓ newname ↓ old name ↓ Table name.

(2) $\rho_{\text{stud}}(B_1, B_2, B_n)$ (R) → old Relation name

↓ new Relation name ↓ new Attributes name.

(OR)

(1) ρ_S (R) — old relation name [If we don't want to change Attributes name]

↓ new Relation name

(OR)

(2) $\rho_{(B_1, B_2, B_n)}$ (R) If we want to change only the attributes names not relation name.

Relational Algebra

Theoritical language → base lang for query language

Relational model — To Retrieve the data in two ways.

Procedural Theoretical
← * Relational algebra
* Relational calculus.

Practical
* SQL is based on Relational Algebra.

* Input ← Relation (tables)
Output will also be in the form of Relation).

Relational Algebra

Unary Operations
* Selection (σ)
* projection (Π)
* Rename (ρ)

Set theory Operations
Union (∪)
Intersection (∩)
Cross product (×)
Minus / Set difference (−)

Binary Op
Joins (⋈)
→ Inner joins
→ Outer joins
→ Left outer join
→ Right outer join
→ Full outer join
→ Natural join
Division (÷)
Assign (←)

Extended or Derived operation
* Aggregate functions

Selection - To Get all records from relation

$\sigma_{(Student)}$

R No	Name	Percen

* Horizontal subset

Student

Rename

Ex: $\rho_{final\text{-}student}(Student)$

Set operations

Input → two relations (tables) After applying set operations

Output → 1 relation ←

Course R

Cid	Cname
101	C++
102	DBMS
103	Python

S

Cid	Cname
101	C++
104	Java
105	C

① Union:- R ∪ S

$\sigma_R \cup \sigma_S$

⇒ All the tuples of R & S.

⇒ Automatically remove the duplicates.

② Intersection: R ∩ S

③ Set difference: R − S (removes all the common attributes)

⇒ It will contain all the tuples of Relation R but not in S.

S − R

⇒ It will contain all the tuples of Relation S but not in R.

④ Cartesian product (R × S)

R

Cid	Cname
101	C++
102	DBMS
103	Python

S

Sid	Sname
S_1	Suresh
S_2	Sarathy

⇒ R×S ⇒ Every tuple of R is associated with every tuple of S.

R×S = 3×2 = 6 Rows.

Cid	Cname	Sid	Sname
101	C++	S_1	Suresh
101	C++	S_2	Sarathy
102	DBMS	S_1	Suresh
102	DBMS	S_2	Sarathy
103	Python	S_1	Suresh
103	Python	S_2	Sarathy

Joins (⋈) = Cartesian product + selection operation

⇒ It combines R_1 and R_2 with respect to a conditions

Join combines tuples of two relations if and only if condition is satisfied. at least one ⇒ Common attribute must be there.

Joins

- Inner Join
  - Theta join: ⋈ condition (Ex: Cartesian product)
  - Natural join / Equi join (*)
- Outer Join
  - Left outer join ⟕
  - Right outer join ⟖
  - full outer join ⟗

1. Theta Join: The matching records will be displayed.

Two relations will be joined based on the Condition. If we join R_1 and R_2 other than the equal condition. non-equi joins

⋈θ Comparison operator ($>, >=, <, <=, \neq$)

$R \bowtie_{A_1 \theta B_1} S$

$R \bowtie_{A_1 = B_1}$, $A_1 < B_1$, $A_1 > B_1$

C

Car model	Price
Car 1	4,00,000
Car 2	5,00,000
Car 2	8,00,000

J.

Jeep model	Price
Jeep1	6,00,000
Jeep2	10,00,000

$C \bowtie_{C\text{-price} < J\text{-price}}$ J-price > C-price

(i) 6,000 00 > 4,00,000 - true.
(ii) 6,000,00 > 5,00,000 - true
(iii) 6,000,00 > 8,00000 - false.

resultant relation

Jeep model	price	car model	price
Jeep1	6,00,000	car1	4,00000
Jeep1	6,00,00	car 2	5,00000
Jeep2	10,000	car1	4,00,000
Jeep2	10,000	car 2	5,00,000

(*) Natural Join: Relations will get joined with respect to common attributes (ie) on equal condition

⇒ Generally join is referred to as natural join

S ⋈ D.

Common attributes name should be same name.

⇒ Values of number of common attribute should be equal.

S

SID	Sname	Dep id
101	Raju	1
102	Ravi	2
103	Ramu	1
104	Hari	3
105	Sonu	5

D.

Depid	Dname
1	CSE
2	ECE
3	EEE
4	Mech.

SID	Sname	DepId	Dname
101	Raju	1	CSE
102	Ravi	2	ECE
103	Ramu	1	CSE
104	Hari	3	EEE

find the employee names who is working in a dept?

Emp

Eno	E-name	Address

Dept

DepNo	name	Eno
D1	HR	1
D2	IT	2
D3	market	4

Outer Joins

1. Left outer Join

S ⟕ D.

⇒ It will Consider all the tuples of S and Common tuples of D of R2

⇒ The mismatching values are filled with NULL

S

SID	SName	DeptID
101	Raju	1
102	Ravi	2
103	Ramu	1
104	Hari	3
105	Suresh	5

D

DeptID	DName
1	CSE
2	ECE
3	EEE
4	mech

SID	SName	DeptID	DName
101	Raju	1	CSE
102	Ravi	2	ECE
103	Ramu	1	CSE
104	Hari	3	EEE
105	Suresh	5	NULL

Right outer Join

S ⟖ D.

⇒ It will Consider all the tuples of D and Common tuples of S

SID	SName	DeptId	DName
101	Raju	1	CSE
102	Ravi	2	ECE
104	Hari	3	EEE
NULL	NULL	4	mech
103	Ramu	1	CSE

Full outer Join → Returns all the tuples when there is a match, Returns null when there is no match.

S ⟗ D.

SID	SName	DeptId	DName

Equi Join

$$R_1 \bowtie_{R_1.R_{no} = R_2.R_{no}} R_2$$

It produces the resultant table with tuples whose Roll no is same in both R_1 and R_2.

Division operation ($\div$) / Derived operator

(i) TO retreive the employee id of the employes working on all projects.

queries containing — Every → use division operation keywords.

TO find employee work on all the projects.

emp

R_1	
Eid	PID
1001	1
1002	1
1002	2
1003	2

R_2	
project	pid
	1
	2

result

Eid
1002

R_2 C R_1.

proper subset, then only division operation can be performed.

Proper subset means every attribute of R_2 must present in R_1 as well as R_2 is not equal to R_1.

R_1 students

Name	Course
Ramesh	DBMS
mahesh	DBMS
naresh	DBMS
Ramesh	DS
naresh	DS
mahesh	CO
Ramesh	CO

R_2 Course

DBMS
DS
CO

2 course
2 count

Here R_2 is a proper subset of R_1. so we can apply division

$R_1 \div R_2$ → returns attributes which is present in R_1 but NOT in R_2. So it contains

R_1 Student Course

name
Ramesh

which ~~name~~ tuples present in name is, the tuple from R_1 which is associate with every tuple of R_2.

R_2
Student
Course

name, course / Course

Assignment operator :- ←

If we have complex queries to perform division like operation, instead of writing it so simple queries & store it in temporary variable

reverse arrow.

temp1 ← query1 res ← temp1 / temp2

temp2 ← query2.

Division operator rules:- (Derived operator)

* Division operator $A \div B$ can be applied if and only if:

1. Attributes of B is proper subset of Attributes of A.
2. The relation returned by division operator will have attributes = (All attributes of A − All attributes of B)
3. The relation returned by division operator will return those tuples from relation A which are associated to every B's tuple.

Student

ID no	sports
101	Class
102	Hockey
102	Class
104	Class

sports
Class
Hockey

student + sports detail

ID no
102

Ex:- Retreive the sid of students who enrolled in every course.

Student

Sid	Cid
S_1	C_1
S_2	C_1
S_1	C_2
S_3	C_2

Course

Cid
C_1
C_2

$$\frac{A(x,y)}{B(y)} = \frac{\text{Student (sid, cid)}}{\text{Course (cid)}}$$

Equi join

Find the Emp name who working in a department having location same as their Address?

Emp

Eno	E-name	Address
1	Ram	Delhi
2	Varun	Chandigarh
3	Ravi	Old
4	Amrit	Delhi

Location

Deptno	location	Eno
D_1	Delhi	1
D_2	pune	2
D_3	patna	4

Add Location

1 Ram Delhi D_1 Delhi 1 Emp ⋈ Location

1 Ram Delhi D_2 pune 2 Emp.Address = Location.Location

2.

Select ename from emp

Inner Join:-

In an Inner Join, only those tuple that satisfy the matching criteria are included, while the rest are excluded.

⇒ Theta Join (θ)

also known as conditional join.

⇒ used when you want to join two or more relation based on some conditions.

Ex:-

$P \bowtie_\theta Q$

Customer

Cid	Cname	Age
101	Amrit	20
102	Vijay	19
103	Sheela	21

Order

Oid	Oname
101	pizza
101	noodles
103	Burger

Customer $\bowtie_{\text{customer.id} > \text{order.id}}$ Order.

Cid	Cname	Age	Oid	Oname
102	Vijay	19	101	pizza
102	Vijay	19	101	noodles
103	Sheela	21	101	pizza
103	Sheela	21	101	noodles

(Customer X order)

$\sigma_{\text{customer.id} > \text{order.Oid}}$ (Customer x order)

Equi join ($\bowtie$)

⇒ Equi join is a special case of conditional join

⇒ when a theta join uses only equivalence condition, it becomes a equi join

⇒ As values of two attributes will be equal in result of equijoin only one attribute will be appeared in result

Customer $\bowtie_{\text{Customer.Cid} = \text{order.Oid}}$ order

Cid	Cname	Age	Oname
101	Amrit	20	pizza
101	Amrit	20	noodles
103	Sheela	21	Burger

Table A

Table B

Natural join

- Natural join can only be performed if there is a common attribute between the relations

2) The name and type of the attribute must be same.

3) Natural join does not use any comparison operator.

4) It does not concatenate the way a cartesian product does.

5) Natural join will also return the similar attributes only once as their value will be same in resulting relation.

6) Only the matching records in both the relation will be displayed

Outer join

- In an outer join, along with the tuples that satisfy the matching criteria, we also include some or all tuples that do not match the criteria.

Left join (⟕)

1) This join returns all the rows of the table on the left side of the join and matching rows for the table on the right side of the join.

2) The rows for which there is no matching row on right side, the result -set will contain null.

A

Id	name
10	Jai
20	veer
30	John

B

Name	marks
Rohan	20
Veer	18
John	14
Sam	13

A ⟕ B

Id	name	marks
10	Jai	NULL
20	veer	18
30	John	14

Right join (⟖)

⇒ This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of join

⇒ The rows for which there is no matching row on the left side, the result will set to null.

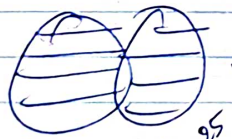
A ⟗ B.

id	name	marks
NULL	Rohan	20
20	Veer	18
30	John	14
NULL	Sam	13

Full-Join (⟗)

- Full outer join creates the result-set by combining result of both left join and right join.
- * The result table will contain all the rows from both the tables.
- * the rows for which there is no matching, the result will contain null values

A ⟗ B.



Id	name	marks
10	Jay	NULL
20	Veer	18
30	John	14
NULL	Rohan	20.
Null	Sam	13

before 21st Th
21st → male
22nd → female
} C.S.

Javatpoint, Tutorialpoint: Advantages & disadvantages of DBMS is nice.

Database → organised collection of data. Database is designed to manage large bodies of Info.

⇒ A database is usually controlled by a database management system. Created

⇒ DBMS is a software that is used to manage the database. (read, update, delete).

Ex: MySql, Oracle, are very popular, Commercial database.

→ The management of data involves both storage of Information & the manipulation of Info.

Graphical User Interfaces, applic program Interfaces.

Adv ① Controls database redundancy - belo it stores all the data in one single datab file

② Data sharing - The authorized users of an organization can share the data among multi users.

③ Easily maintainence - centralized