

Relational Data Base Management System.

Unit - I

Introduction: View of Data - Database language - Relational database - Data base design - Transaction Management - Data Architecture - Introduction to the Relational Model: Structure of Relational Databases - Databases Schema - keys - Schema Diagrams - Relational Query Languages - Relational Operations.

Unit - II

Introduction to SQL: Overview of the SQL Query Language - SQL Data Definition - Basic Structure of SQL Queries - Additional Basic Operations - Set Operations - NULL Values - Aggregate Functions - Nested Sub Queries - Modification of the Databases. Intermediate SQL: join expressions - Views - Transactions - Integrity Constraints - SQL Datatypes and Schemas.

Unit - III

Advanced SQL: Functions and Procedures - Triggers - Formal Relational Query Languages: The Relation Algebra - The Tuple Relational Calculus - The Domain

Unit - IV

Relational Database Design - Atomic Domains and First Normal Form - Decomposition using Functional Dependencies - Functional Dependency Theory - Decomposition using Multivalued Dependencies - More Normal Forms, Transaction Management : Transactions - Transaction concept - Serializability.

Unit - V

Error Handling: Exception - Built-in Exceptions - User Defined Exceptions - Subprograms in PL/SQL : Describing a Subprogram - Procedure - Function - Trigger : ~~Trans~~ Sample Trigger - Trigger Concept

1. Table Manipulation.

Table Creation, Renaming a Table,
Copying another Table, Dropping a Table.

2. Table Description:

Describing Table Definitions Modifying
Tables, joining Tables, Number and Date
functions.

SQL Queries and Sub Queries.3. SQL Queries: Queries, Sub Queries, and
aggregate function.4. DDL: Experiments using database DDL
SQL statement.5. DML Experiments using database DML
SQL statement6. DCL Experiments using database
DCL SQL statement.Exception Handling and PL/SQL7. Exception Handling, PL/SQL procedure
for application using exception handling8. Functions: PL/SQL procedure for applica-
-tion using function.9. Cursor: PL/SQL procedure for application
using cursor.

10. Trigger: PL/SQL procedure for application using trigger.

11. Package: PL/SQL procedure for application using package.

1. What is DBMS?

It is a collection of data and Management system ~~is~~ ^{is a} set of programs to store and retrieve those data.

Definition

"DBMS is collection of interrelated data and set of programs to store and access those data in an easy & effective manner".

2. What is need of DBMS?

It is basically developed for large amount of data

↓

two thing

i) storage of data

ii) retrieval of data

i). Storage of data.

data is stored in such a way that it acquires lot less space as the redundant data has been removed before storage.

retrieval of data.

Since data is stored in an optimized & systematic manner we can access quickly when need

Data Base Application

- * Telecom
- * Banker
- * Sales
- * Airlines
- * Education
- * Online shopping

Advantage of DBMs $\leftrightarrow$ (file processing system)
removing duplicate copy

i) Data redundancy

often leads to higher storage cost & poor access time.

ii) Data inconsistency

Accuracy of data.

iii) Data isolation

data are stored in various files & it may be in different formats.

iv) Depending on application program

v) Atomicity issues

It refers "all or nothing" every database

but nothing
either all the transactions operations are executed or none will be done
changing files would lead to change in application.

iv
Define ←

View of Data in DBMS

→ Abstraction is one of the main features of database systems

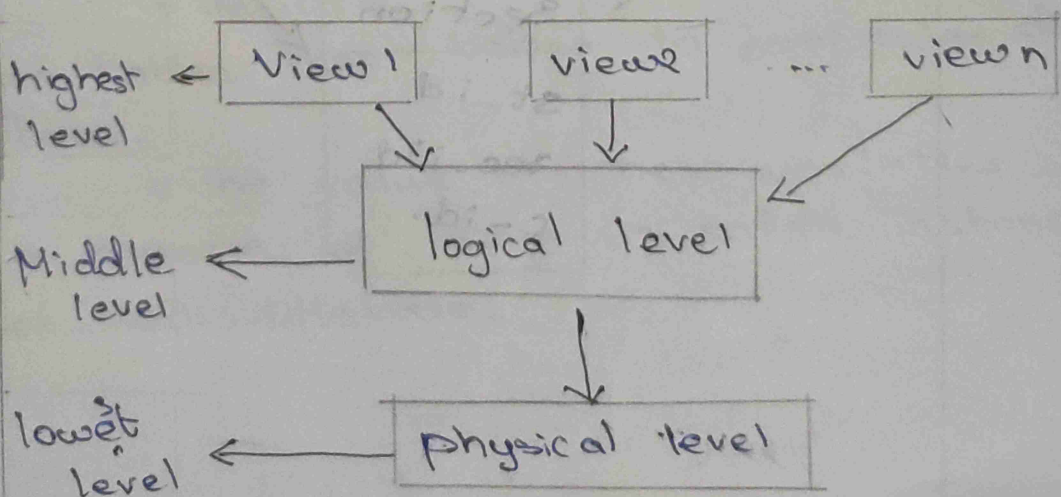
Hiding irrelevant details from user & providing abstract view of data to users helps in easy & efficient user.

Database interaction in different levels of views such as.

- * Data relationship
- * Data base Schema
- * Data constraints
- * Data Security etc.. from user.

1) Data Abstraction

process hiding irrelevant details from user is called data Abstraction.



① Level → It describes the user interaction with data base system

② Level → It is the middle level of data abstraction architecture. It describes what data is store in database.

③ level → which describes how data is actually stored in database.

q.1.22

DBMS Schema

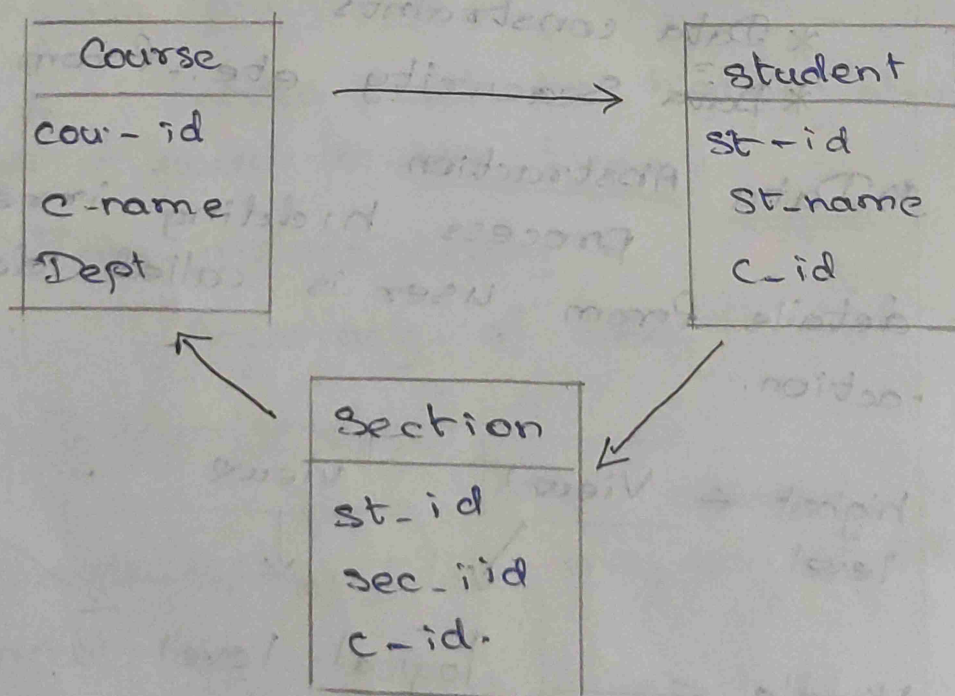
Design of a database is called the Schema. ~~is~~ Schema is of three types

i) Physical schema.

ii) logical schema.

iii) view schema.

for. eg.



(a) a design of database at physical level is called physical schema.

(i.e.)

* how data is stored in blocks of storage *

(b) a design of database at logical level is called logical schema.

eg.

programmers & database administrators work at this level.

(c) a design of database at view level is called view schema.

(i.e.)

generally end user interaction with database system.

DBMS Instance

Definition

data stored in database at a particular moment of time is called instance of data base.

Database Schema

* defines the variable declaration in tables that belong to particular database.

* The value of these variables at a moment of time is called instance of that database.

(a) a design of database at physical level is called physical schema.

(i.e.)

* how data is stored in blocks of storage *

(b) a design of database at logical level is called logical schema.

eg.

programmers & database administrators work at this level.

(c) a design of database at view level is called view schema.

(i.e.)

generally end user interaction with database system.

DBMS Instance

Definition

data stored in database at a particular moment of time is called instance of data base.

Database Schema

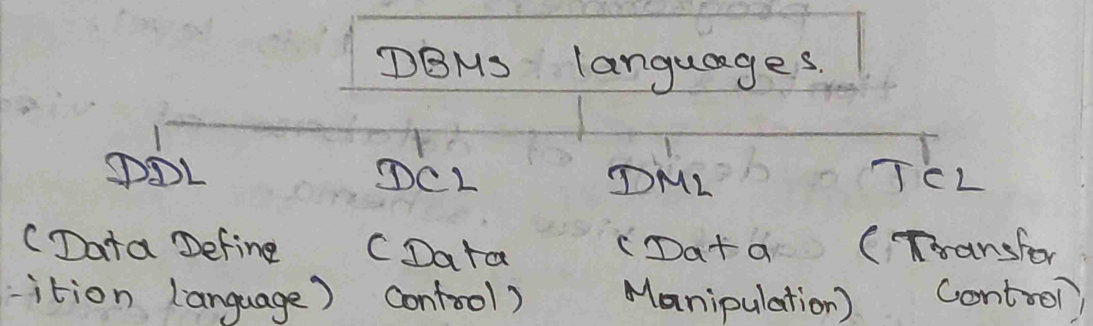
* defines the variable declaration in tables that belong to particular database.

* The value of these variables at a moment of time is called instance of that database.

DBMS language.

These are used read, update & store data in a database. There are several such languages that can be used to this purpose one such is SQL → Structure Query language.

Types of DBMS languages.



• It refers to DBMS language.

* It is used for specifying the database schema.

* It is used for creating tables, schema, indexes, constraints, etc. in database.

* We can perform following operations using DDL.

DDL Commands.

1. Create

5. Rename

2. Alter

6. Drop / column Delete

3. Drop / row Delete

7. Comment

4. Truncate

11.7.22. DCL 6

* It is used for granting & revoking user access on a database

1. To grant access to user - GRANT

2. To revoke access from user - REVOKE

All these are parts of single database language such as SQL.

DML -

* It is used for accessing & manipulating data in a database.

* The following operations on database comes under DML.

1. To read record from table - SELECT

2. To ^{insert} ~~read~~ record from into table - INSERT

3. Update the data in tables - UPDATE

4. Delete all the records from the table - DELETE

TCL -

Changes in the database that we made using DML commands are either performed or roll backed using TCL.

1. To Persist the changes made by DML commands in database - COMMIT.

2. To rollback the changes made to the database - ROLLBACK

Database Design.

Process of designing the general structure of the database

Logical Design - deciding on the schema
It requires good collection of relation schemas

Physical Design - deciding on the physical layout of the database.

→ we need a method to ensure each relation in the database is good
Two way of doing so.

Entity Relationship model.

* Models in organization a collection of entity, Table, and relationships.

* Can be Represent in a diagram. using ER-diagram.

Normalisation.

for malise the needed data in a perfect model

12.7.22 Storage Management

Storage Management is a module that provides interface between the low level data stored in the database and the application programs & queries submitted to the system.

Storage manager is responsible to the following tasks:

- i) Interaction with the OS file Manager
- ii) Efficient storing, retrieving & updating of data.

Issues

- i) Storage access
- ii) File organization
- iii) Indexing & hashing.

1. Transaction Management

A transaction is a collection of operations that performs a single logical function in a database application.

2. Transaction Management Component

It ensures that the database remains in a correct state despite system failures & transaction failures.

3. Concurrency control . Management

It controls the interaction among the concurrency transactions to ensure the consistency of the database.

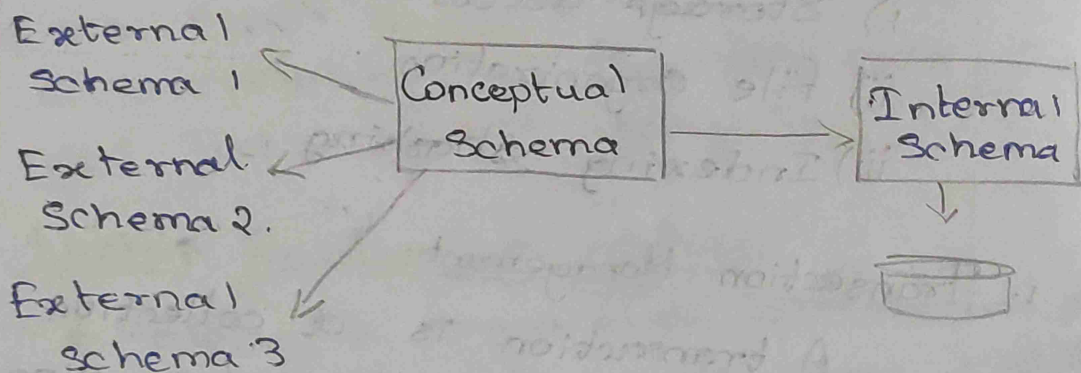
13.7.22.

Database Architecture

Database users

user & Interfaces are

- i) Naive users
- ii) Application programmers
- iii) Sophisticated users
- iv) specialised users



Database Administrator (DBA)

- * Schema definition
- * ^{chema}Storage and physical organization modification
- * Storage structure and access - Method definition
- * Granting of authorization for data access.
- * Routine maintenance.

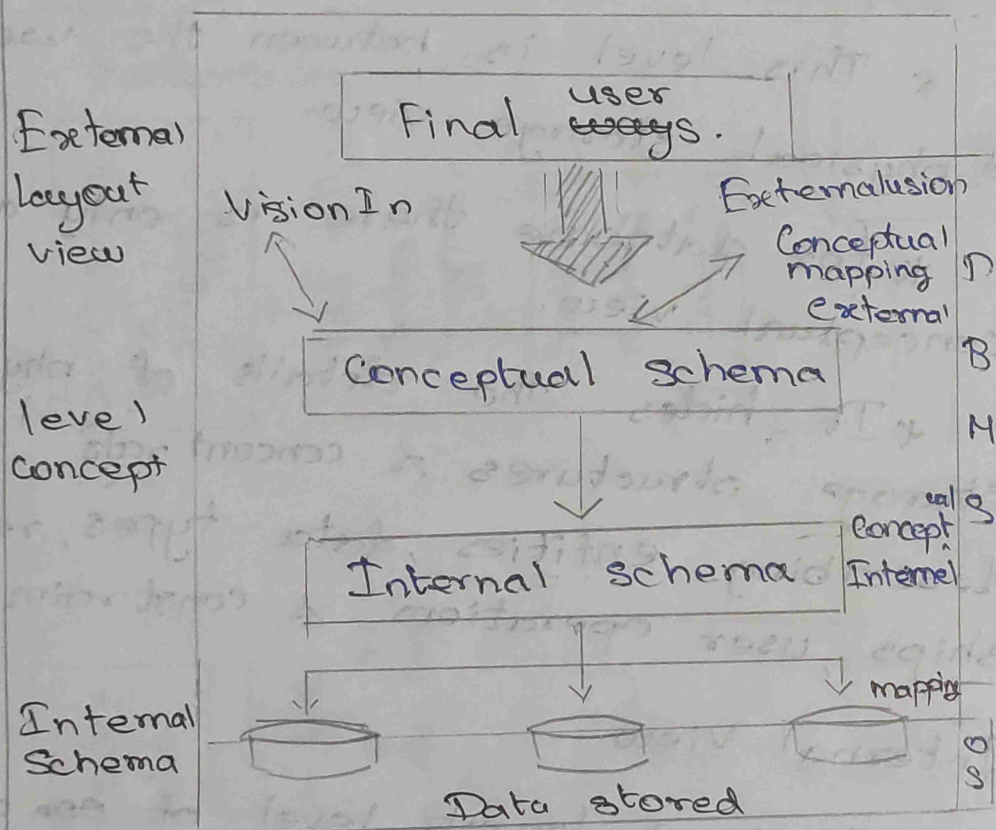
⇒ DBA Manage all levels of DBMS Model

Three levels of Architecture.

* It is also called a schema Architecture

* This framework is used for describing the structure of specific database system.

* here database schemas can be defined at three levels.



External view

* Top level of abstraction and it is concerned with the user.

* Each External schema describes the part of the database that a particular user is interested in and hides the rest of the database from users.

* There can be 'n' no. of external views for database, where 'n' is the

of users

* eg. users of account dept will be interested in fees detail of student instead of personal details.

* All database users work on external level of DBMS.

Conceptual view.

* This level is between the user level & physical storage view.

* for database there is only one conceptual view.

* It hides the details of physical storage structures & concentrates on describing entities, data types, relationships, user operations & constraints

Internal View.

* It is a lowest level of data abstraction

* It is the information about the actual representation of entire database

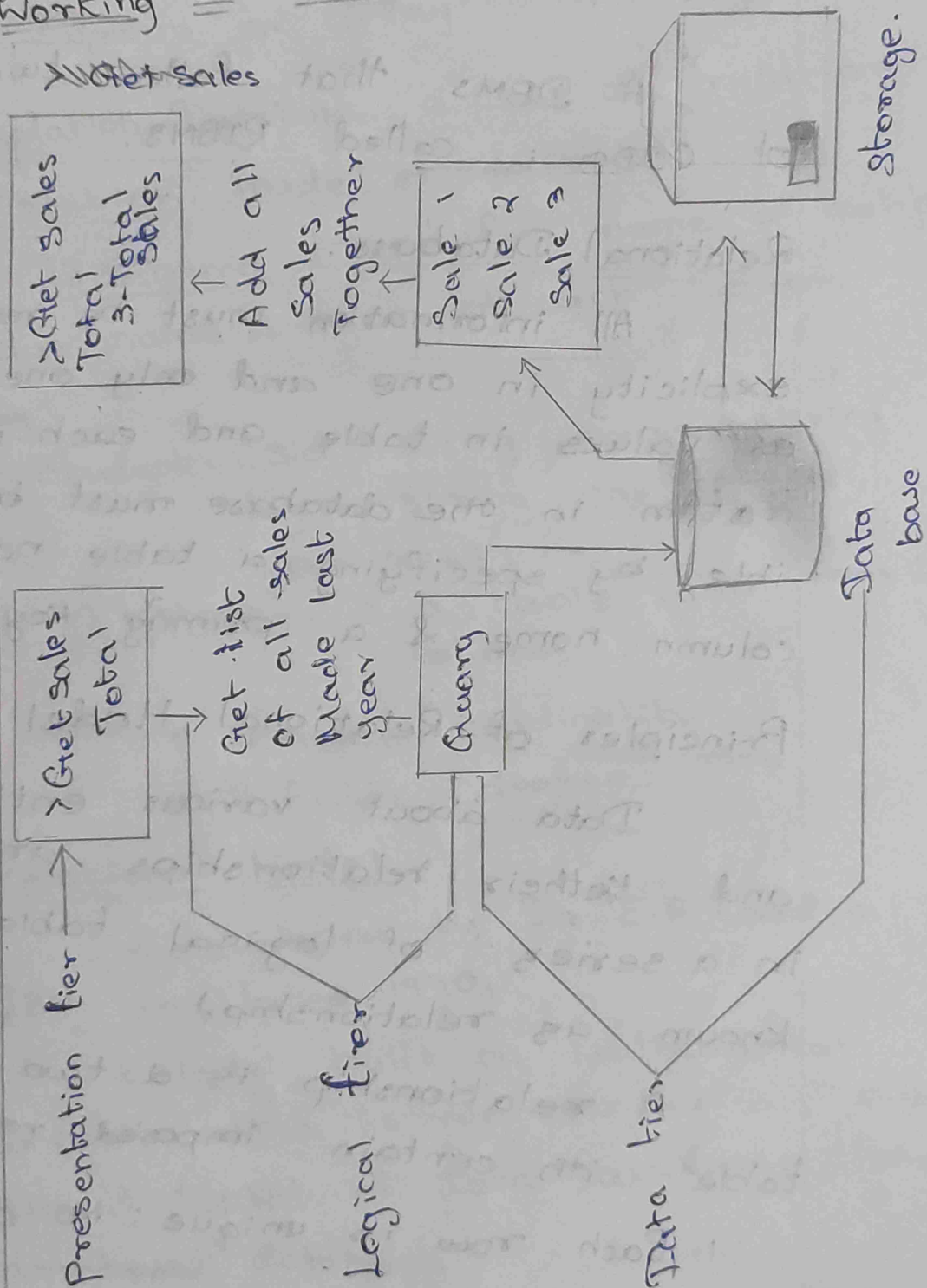
* It is a closer to the physical storage method

* This is the view that tells as even what data is stored database and how it considers the aspects of storage

allocation, Access paths ~~etc~~ etc.

* The Internal view does not deal with physical devices directly. Instead it view physical devices as a collection of physical page and allocates space in terms of logical pages

Working of three level architecture



18.7.22.

~~18/7/22~~

RDBMS:

Definition.

"A DBMS that manages data as collection of tables in which all data relationships are represented by common values in related tables".

"A DBMS that follow twelve rules of CODD is called RDBMS".

Relational Database.

All information must be represented explicitly in one and only one way: as values in table and each & every datum in the database must be accessible by specifying a table name, a column name & a primary key.

Principles of Relational Model

Data about various entities and their relationships are stored in a series of logical tables (also known as relationship.)

A relationship is a two dimensional table with certain imposed restriction.

1. Each row is unique: No duplicate row.

2. Entities in any column have the same Domain.
3. Each Column has a unique name.
4. Order of the columns or rows are irrelevant.
5. Each entry in the table is single valued. No group item, repeating group or array is allowed.

Relation Database: a set of relations.

Relation: made of 2 parts.

i) Schema → specifies name of relation plus name & types of each column.

eg:

Student: (Sid : string, name : string,
login : string, age : integer,
gpa : real)

ii) Instance → a table with rows & columns.

* # rows = cardinality

* # fields = degree.

Relational Model

* developed by Dr. E.F. Codd at IBM in late 1970s

* Model built on Mathematical concepts, which is represented a "Relational Model of Data for large shared databases" databanks

At core.

It is a concept of a table in which a data is stored records & fields are there.

Table (Relation)

eg.

Supplier

prime
key
a

S. Name

Domain
name
Quantity

Domain
name
City.

Tuples	S ₁	palprandi	20	Ahmedabad
	S ₂	Karthi	10	Surat
	S ₃	Lingesh	40	Baroda
	S ₄	Aathi	35	Madurai
	S ₅	Viresh	25	Chennai

↑
↓ cardinality

Attributes
— or —
Degree

25/7/22 Relational Model

Advantages

- * The data access methodology in Relational model is quite different from & better than the earlier database model
- * It provides useful tools for database administration.
- * Meta-data (data about table & field name).
- * It can use operations recursively in order to provide information about the database.

i) Attribute - Member of relation type.

ii) Attribute name - All attribute names must be unique within a table & relation.

iii) Attribute Domain

iv) Attribute value set

v) Relation / Table Degree.

vi) Tuples

vii) Cardinality.

iii) A set of all possible values that can be attained by an attribute.

iv) Values currently contained in an attribute.

v) No. of attributes in a relation/table

vi) Rows in a table/relation

vii) No. of tuples in a relation.

Ex.

Instance of students Relation.

Student (Std.no, name, address) $\rightarrow$ Schema

Student (123, Kavi, Madurai) $\rightarrow$ Instance.

Stdno	name	address	} cardinality of this relation is
123	Kavi	Madurai	
124	Raj	Tirichy	
125	Prakash	Madurai	

Relational Model

Database Schema.

description of database is called database schema. A database schema is description during database design & not expected to change frequently.

Schema diagram.

Display schema called schema diagram. each object in schema is a schema construct.

Database Instance

The data in a database of a particular moment of time.

Intension & Extension

The schema is sometimes called the instance and a database instance is called an extension of the schema.

Basic components of a Relation.

Attributes (Columns)

cust-id	cust-name	Region	status.

Tuples
rows

Intersection of a
column & a row is a field
that contains a value

Key in DBMS

key ∴ A key is a set of attributes that can identify each tuple uniquely in the given relation.
* There are various types of keys in DBMS.

→ Candidate key:

* A set of minimal attributes that can identify each tuple (row) unique in the given relation is called as a candidate key.

* It is possible to have multiple Candidate key.

eg: student (roll; name, age, section, class
address; gender);

In these candidate key,

{ roll, class, section }

Keys

→ Candidate key

→ Primary key

→ Super key

→ Foreign key

→ Secondary key

→ unique key

→ Alternate key

→ composite key

Partial key

Surrogate Key.

ii) (name, address) } These are candidate key because each set consists of minimal attributes required to identify each student uniquely in that table.

26/7/22

Super key

* It is a set of attributes that can identify each tuple uniquely in the given relation.

* It is not restricted to have any specific no of attributes.

* Thus, a super key may consist of any number of attributes.

eg:

* Student (roll, name, age, address, class, section)

* (roll, name, age, class, address)

* (class, section, roll)

* (name, address)

Primary key:

It is a candidate key that the database designer selects while designing the database.

Foreign key:

* An attribute 'x' is called as a foreign key to some other attribute 'y' when its values are dependend on the values of attribute 'y'

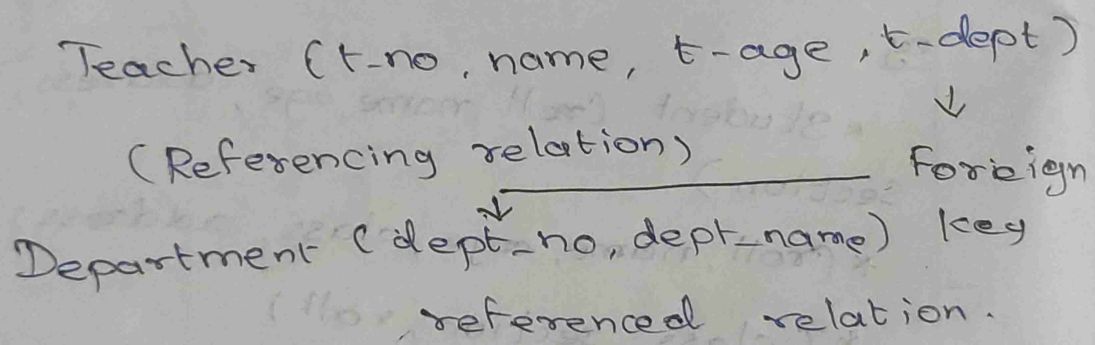
* The attribute 'x' can assume only those values which are assumed by the attributes 'y'

* Here the relation in which attribute 'y' is present is called as the referenced relation.

* The relation in which attribute 'x' is present is called as the referencing relation.

* The attribute 'y' might be present in the same table or in some other table

eg.



Partial Key.

* It is a key using which all the records of the table can not be identified uniquely.

* A bunch of related tuples can be selected from the table using the partial key

eg.

Department (Empno, Dependent, name, Relation)

Empno	Depname	Relation
E1	xx	Mother
E2	yy	Father

Composite key

* A primary key comprising of multiple attribute and not just a single attribute is called as a composite key.

Unique key

* It is key with the properties like
 * It is unique for all the records of the table

* Once assigned, its values can not be changed (ie) it is not updatable.

* It may have a NULL value

eg: best eg is adhar no.

Secondary key

It is a ~~require~~^{used} for the indexing purpose for better and faster searching.

21/7/22

Query languages in DBMS

* Query language is a language which is used to retrieve information from a database.

* It is divided into two types

i) procedural language.

ii) Non procedural language.

i) Procedural language.

Information is retrieved from the database by specifying the sequence of operations to be performed. eg: All relational algebra types are there.

ii) Non procedural language.

Information is retrieved from the database without specifying the sequence of operations to be performed. users specify only what information is to be ~~retrieved~~ retrieved.

eg: Relational calculus two types

i) Tuple calculus - Row

ii) Domain calculus - column.

Query operations.

All relational Algebra operations comes under Query operations They are.

i) Select operation

ii) project operation.

iii) Rename operation

iv) Union operation

v) Intersection operation.

vi) Different operation

vii) Cartesian product operation

viii) Join operation.

ix) Division operation.

Union Operation:

Combines values in R_1, R_2 by removing duplicate only

Syntax: $R_1 \cup R_2$

Project Operation.

It Display specific column of a table. It is denoted by π It is a vertical subset of the original relation. It eliminates duplicate tuples.

Syntax: $\pi_{\text{regno}}(\text{student})$

Intersection operation.

It Display the common values in $R_1 \& R_2$. It is denoted by $\cap$

Syntax: $R_1 \cap R_2$

eg: $\text{regno}(R_1) \cap \text{Regno}(R_2)$

Division operation.

The division operation is used for queries which involve the "all"

$R_1 \div R_2$ = tuple of R_1 associated with all tuples of R_2 .

eg: retrieve the name of subject taught in all courses

R_1	<table><tr><th>Name</th><th>Course</th></tr><tr><td>System</td><td>B.tech</td></tr><tr><td>Database</td><td>M.tech</td></tr><tr><td>Database</td><td>B.tech</td></tr><tr><td>Algebra</td><td>B.tech</td></tr></table>	Name	Course	System	B.tech	Database	M.tech	Database	B.tech	Algebra	B.tech
Name	Course										
System	B.tech										
Database	M.tech										
Database	B.tech										
Algebra	B.tech										

R_2	Course
$\frac{2}{1} \cdot$	B.tech
	M.tech

	<table><tr><th>Name</th></tr><tr><td>database</td></tr></table>	Name	database
Name			
database			

o/p.

28.7.22

Operations in DBMS

Cartesian product operation.

* It combines R_1 & R_2 without any condition. It is denoted by $\times$. Degree of $R_1 \times R_2$ = degree of R_1 + degree of R_2 .

(degree = total no. of columns)

eg: R_1

Regno	Branch	section
1	CSE	A
2	ECE	B
3	civil	C

R_2

Name	Reg no.
Banu	2
Raja	3

B No	Branch	Section	Name	Reg. no
1	CSE	A	Banu	2
1	CSE	A	Raja	3
2	ECE	B	Banu	2
2	ECE	B	Raja	3
3	civil	C	Banu	2
3	civil	C	Raja	3

Select Operation.

It displays the records that satisfy a condition. It is denoted by sigma (σ) and is a horizontal subset of the original relation.

Syntax.

σ condition (table name)

eg:

take student table.

* To display all records.

σ (student)

* To display all records of CSE branch.

σ branch = CSE (student)

* Particular reg. no.

σ Regno = 2 (student)

* To check two condition.

i) σ branch = ECE A Section = A (student)

ii) σ Section = B $\cap$ Branch = CSE $\cap$ V branch
= civil (students)

Har eg.

eg: σ salary > 20000 $\wedge$ designation = HOD $\wedge$
dept no = 2 (emp)

Rename Operation.

It is used to assign a new name to a relation. It is denoted by

ρ (rho)

Syntax:

ρ newname (table name)

Rename can be done for a

* relation

* attribute

* both

Different operation.

* The different between two relations (R_1, R_2); here all rows from the first relation that are not present in the secondary relation.

* This operation also relies on the two tables being union compatible.

Person R_1

Name	age	food
John	15	juice
Jiva	32	chip
Raja	27	Cerry

R_2 Person

Name	age	food.
John	15	juice
Raja	27	Cerry

new $R_1 - R_2$

Name	age	food
Jiva	32	chip.

Join Operation.

Takes two or more table and combines them into one table there are several types of join command, They are.

- * Natural join.
 - * Equi join
 - * Theta join
- } Inner join

join operation denoted by ' $\bowtie$ '

- * Left outer join
 - * Right outer join
 - * full outer join
- } outer joins.

Inner join

In this join only those tuples that satisfy the matching criteria are included, while the rest are executed.

Theta join:

general case of join operation is called a Theta join. It is denoted by θ .

$A \bowtie_{\theta} B$. These can use any condition in the select criteria.

eg

$A \bowtie_{A \text{ column } 2 > B \text{ column } 2(B)}$

column 1	column 2
1	2.

Equi join:

when a theta join uses only equivalence condition; it becomes an equi join.

eg:

$A \bowtie_{A \text{ column } 2 = B \text{ column } 2(B)}$

column 1	column 2.
1	2 1

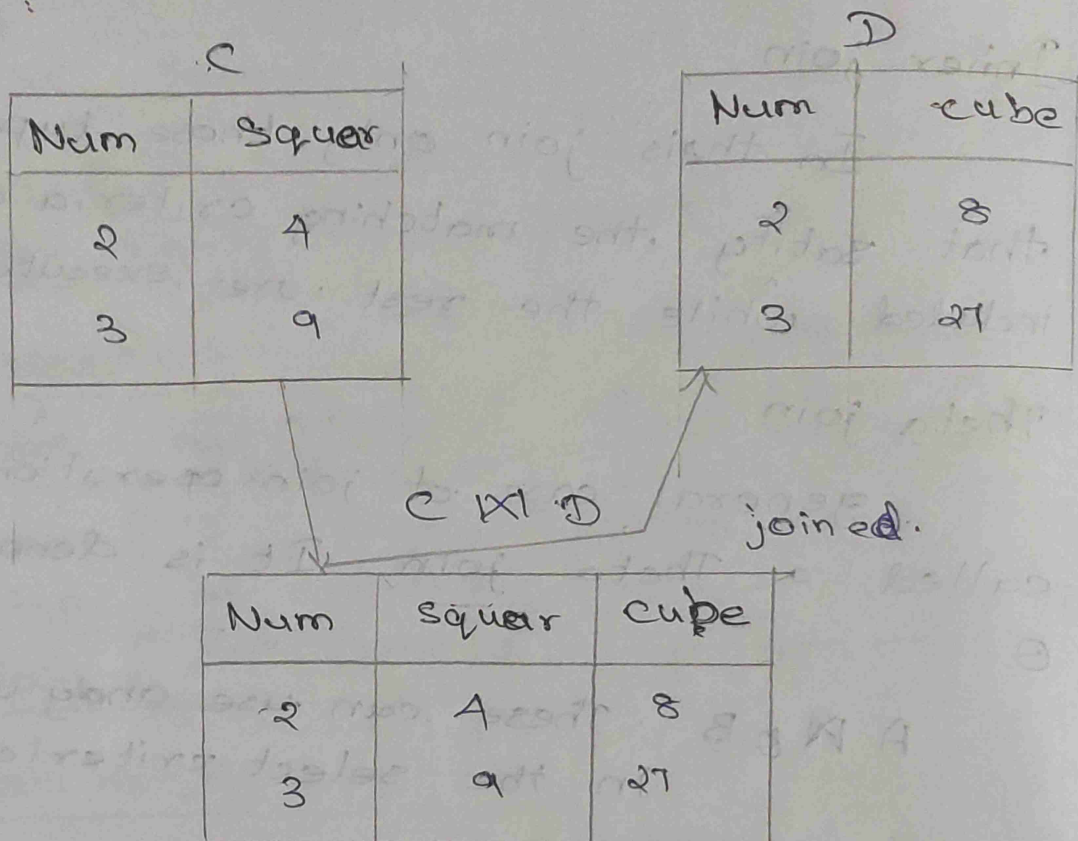
Natural join

It can be performed if there is a

Common attribute (Column) between relation
name & type of an attribute must be

Semk:

eg:



1.8.22.

Unit - II . SQL - Structured Query Language

Introduction.

* IBM sequel language developed as part of system 'R' project at IBM.

* Renamed as SQL

* ANSI X.3.130 : Standard SQL.

⇒ SQL - 86

⇒ SQL - 89

⇒ SQL - 92

⇒ SQL : 1990

⇒ SQL - 2003

* Commercial system offers most, of all SQL - 92.

* Plus varying features sets from later stds & special features.

DDL: SQL DDL allow specifications of information about relations, including.

* Schema for each relations.

* domain values associated with each attribute.

* Integrity constraints.

* And other information like

⇒ set of indices to be maintained for each relations

⇒ Security and authorization information for each relation

⇒ physical storage structure at relation on disk.

Domain types in SQL.

- char(n)
- varchar(n)
- int
- Small int
- numeric (p,d)
- real, double precision
- float(n)

Basic Query structure.

* A typical SQL Query has the form:

Select $A_1, A_2, \dots, A_n \rightarrow$ Attributes.

from $r_1, r_2, \dots, r_m \rightarrow$ relations

where $P \rightarrow$ is a predicate.

The result of an SQL Query is a relation.

The select clause:

* The select clause lists the attributes derived in the results as a Query. • correspond to the projection operation as the relational algebra.

Fig

find the names of all instructor

select Name

from instructor

Note SQL Name are case insensitive

Eg. Name = Name = Name.

2/8/20

Select clause.

* An asterisk in the select clause denotes all attributes.

↓
Select * from Instructor

An attribute can be a literal with no "From" clause.

Select '437' → result a single row with value 437.

Select clause can contain arithmetic expressions involving the operation +, -, %, *, / and operating on constants are attributes of tuple.

The query.

Select ID, name, salary / 12 from instructor.

here it list all attributes as in table but salary alone will be divided by 12 will be resulted.

where clause.

* This clause specifies conditions that the result must satisfy the given condition. * corresponds to the selection predicate of the relational algebra.

* To find all instructions in comp sci. dept ; select name from instructor where dept.name = "comp.sci";

* Comparison results can be combined using the logical connectives 'and', 'or' and 'not.'

* To find all instructors in comp.sci with salary > 80000

Select name from instructor,
where deptname = 'comp.sci' and salary
= '80000'

* Comparisons can be applied to the results of arithmetic expressions.

From clause:

$\Rightarrow$ This clause lists the relations involved in the query.

* corresponds to the cartesian product operation of the relational algebra.

$\Rightarrow$ Find the cartesian product in instructor $\times$ teaches.

Select * from instructor, teacher;

$\hookrightarrow$ This results all the possible instructor-teacher pair from both relations.

cartesian product is useful when combined with where clause condition.

eg. ① find all instructor who have taught some course and the course-id.

$\Rightarrow$ Select name, course-id
from instructor, teacher.

where instructor_id = teacher_id.

③ Find the names of all instructors in the art department where have taught some course & course_id

⇒ Select name, course_id

from instructor, teacher

where instructor_id = teacher_id &

- instructor department ~~instructor~~,

name = 'Art'

Set Operations:

⇒ Set operation union, intersect and except

* Each of the above operations automatically eliminates duplicate.

* To retain all duplicates use the corresponding multiset various union all, intersect all & except all

If a tuple occurs 'm' times in r and n times in s then it occurs

* m+n times in r union all s

* min(m, n) times in r intersect all s

* max(0, m-n) times in r except all s

NULL values:

⇒ It is possible for tuples to have null values, denoted by null, for some of their attributes.

⇒ null signifies an unknown value or that a value does not exist.

⇒ The result of any arithmetic expression involving null is null.

eg: 5 null returns null.

⇒ The predicate is null can be used to check for null values.

eg: find . all instructors whose salary .
select name
from instructor .
where salary is null ;

⇒ Null values has three valued . logic as
→ True , false , unknown .

True:

'p' is unknown predicates to true if
'p' evaluated to unknown.

false:

Result of where clause predicate is treated as false if it evaluates to unknown.

Unknown:

Any comparison with null returns unknown.

eg: 5 < null or null < 5 or null > null or null = null

Aggregate functions

These functions operate on multiset values of column of a relation & return a values.

avg → average value

min → minimum value

max → maximum value

Sum → sum of value

count → number of value.

Aggregate function - Grouped by

Attributes in select clause outside of aggregate functions must appear in grouped by list.

eg: select deptname avg (salary) as avg. Salary from instructor group by deptname

Aggregate functions - Having clause

Predicates in the having clause are applied after the formation of groups where as predicates in the where clause are applied before forming groups.

eg: find the names & average salaries of all dept whose average salary is greater than 4000.

" select deptname, avg (salary)
from instructor
group by deptname
having avg (salary) > 4000.

Views:

definition

* A view provides a mechanism to hide certain data from the view certain users.

* Any relation that is not of conceptual model but is made visible to a user as "virtual relation" is called a view.

* A view is defined using the create view statement which has the form.

Create view V as $\langle \text{query expression} \rangle$

↓

view
name.

↓

legal SQL
expression.

* Once a view is defined, the view name can be used to refer to the virtual relation.

* view creation is not as general table creation rather it causes the saving of an expression.

eg: Create view faculty as select id, name, deptname from instruction;
a view can be expanded in a query

(as another view).

eg: Create view physics_fail as (select course id, room no from (select course, course id, building, room_no, from course, Section where course.course id = Section.course id and course.deptname = 'physics' and Section.semester = 'fall' and section : year = '2009') where building = 'cs block');

A view expansion of an expression repeats the following steps.

- i) Find any view relation V_i in e_i
- ii) replace the view relation V_i by the expression defining V_i until no more view relation are present in e_i

10.8.2

Transaction:

- * It is unit ^{of} work.
- * Automatic transaction is either executed fully or rolled back as if it never occurred.
- * Transaction being implicitly ended by commit work or rollback work by default must database commits automatically by each SQL statement.

Integrity constraints:

* It is used to guard against accidental damage to database.

* It makes sure, authorized change not results in loss of data consistency.

* Constraints on a single relation are

⇒ not null eg. name varchar(20) not null

⇒ primary key budget numeric(12,2)

⇒ unique

⇒ check (P), where P is a predicate



It is used to check the attributes values.

Unique constraint (A₁, A₂, ..., A_m)

* This states that the attributes A₁, A₂, ..., A_m form a candidate key.



These keys are permitted to be null contrast to primary key.

Referential Integrity

It refers a value in one relation for an attribute in a certain set of attributes in another relation.

eg: let 'A' be a set of attributes & let "R & S" be two relations containing the attribute 'A' & where 'A' is primary key of S & A is said to be a foreign key of R.

Functions & procedures.

* This is supported by SQL 1999.

* Functions & procedures are written in SQL itself or in an external programming language (eg. C, Java)

* Functions written in an external language are particularly useful with specialized data types such as integers & geometric objects

* Some database systems support to valued functions which can return a relation as a result.

* SQL 1999 supports rich set of imperative constructs, including $\rightarrow$ loop, if-then-else assignment.

SQL Functions.

Define a function as below as

eg.

create

The compound statement `begin-end` contains multiple SQL statements
`returns` → indicates variable type that is returned.

`return` → specifies values that are to be returned as a result of invoking function.

In SQL 2003 -- added table functions that return a relation as a result.

eg:

Create function `instructor_of (dept_name char(20))` returns table

`ID varchar(5),
name varchar(20),
dept_name varchar(20),
Salary numeric(8,2)`

return table.

`select ID, name, dept_name, salary, from
instructor where instructor.dept_name =
instructor_of(deptname)`

usage

`select *`

`from table (instructor_of('Music'))`

5-9-20

Modeling

* A Database can be modeling a collection of entities.
→ tables

* relationship among of entities.

An Entity

* An entity is an object that exists and is distinguishable from other objects.
different

* Entities have attributes.

* An Entity set is a set of entities of the same type that share the same properties.

Eg:

Set of all persons, companies, etc

Entity set

Instructor

ID	name
101	xx
102	yy
103	zz

Student

ID	name
S01	a
S02	b
S03	c

Relationship sets

* A relationship is an association among several entities.

Eg: S01 (a) advisor 101 (xx)

Student entity relationship instructor entities

* A relationship set is a mathematical relation among $n \geq 2$ entities, each taken from entities sets.

eg: (901, 101) $\in$ advisor relationship.
Entity sets belongs to advisor relationship.

Relationship sets:

An attributes can also be a property of a relationship set.

For eg: the advisor relationship set between entity sets instructor and student may have the attribute date which tracks where the student being associated with the advisor.

101	xx	3 May 2021	S01	a
102	yy	10 June 2021	S02	b
103	zz	12 June 2019	S03	c

Degree of a relationship set binary relationship:

* It involves two entities sets

* Most relationship between more (or degree two) than two entity sets are rare.

Attributes:

* An entity is represented a set of attributes that is descriptive properties possessed by all members of an entity set:

eg: Instructor (ID, name, street, city, salary)
Course (courseid, credits)

Domain - the set of permitted values for each attributes.

Attribute type:

⇒ Simple and composable attributes.

⇒ Single valued and multi valued attributes

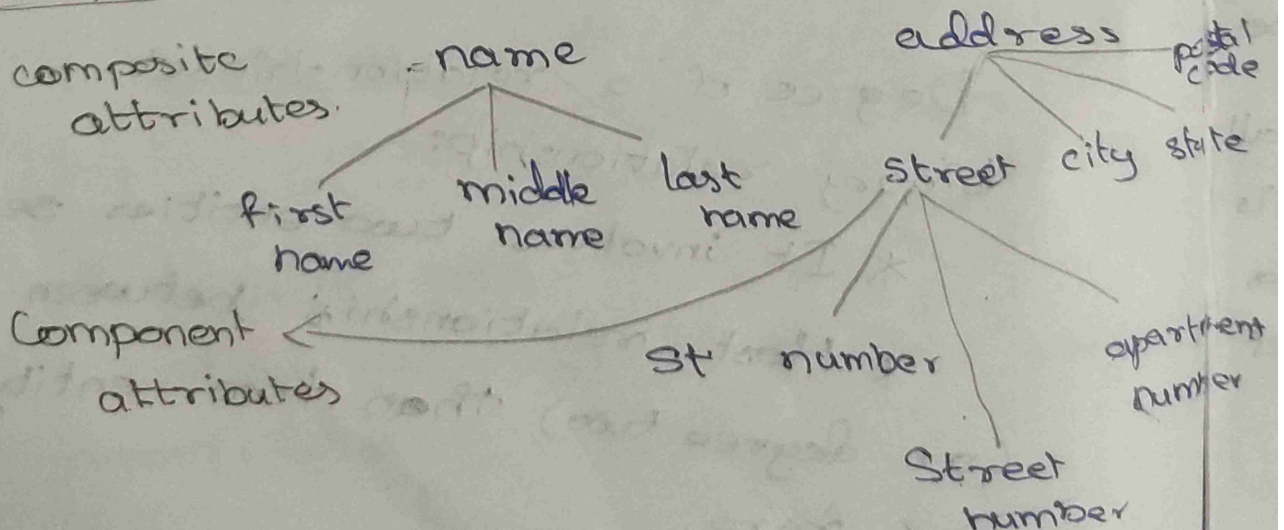
eg: phone numbers.

Derived attributes

* Can be computed from other attributes

eg: age, given date of birth.

Composite attributes.



Mapping Cardinality constraints

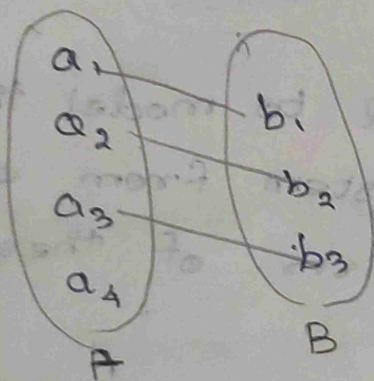
* expresses no of entities to which another entity can be associated via a relationship set.

* More useful in describing binary relationship sets

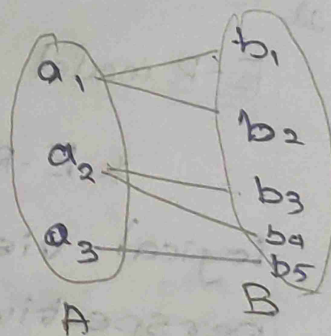
* for a binary relationship set the mapping cardinality must be one of the following types

- ① one to one
- ② one to many
- ③ many to one
- ④ many to many

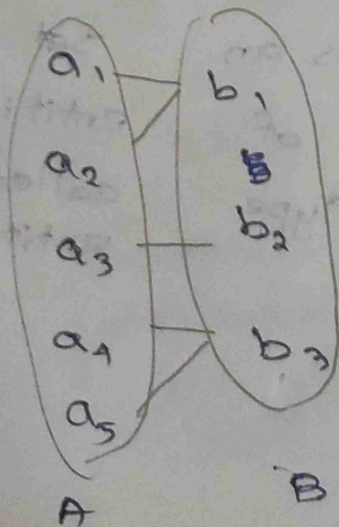
① One to one



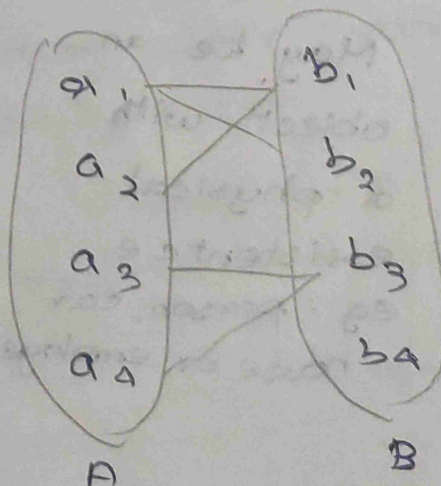
② One to many



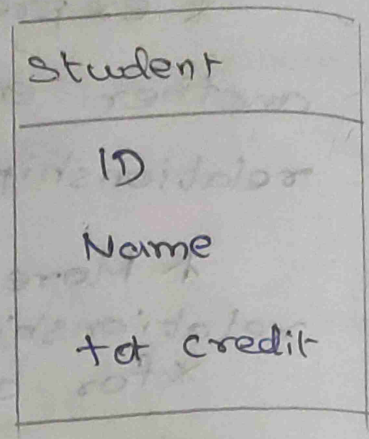
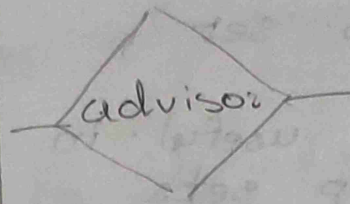
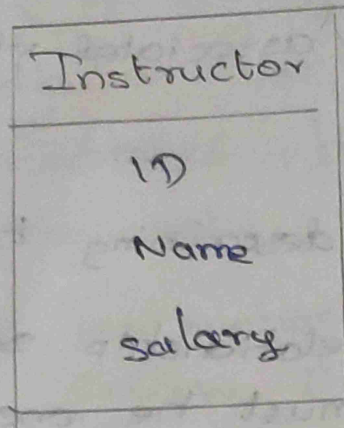
③ Many to one



④ Many to many



E-R Diagrams



- * Rectangles represent entity sets
- * Diamonds represent relationship sets
- * Attributes listed inside entity rectangles
- * underline indicates primary key
- * attributes

8-9-22

E-R model Diagram:

* E-R model is used to model the logical view of the system from data perspective which consists of these components

Entity,



May be an object with a physical existence
eg: person, car, house or employee

Entity Type



Entity is an object of Entity Type

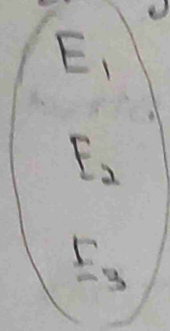
Entity set



Set of all entities is called an entity set.

Student

Entity type



entity set

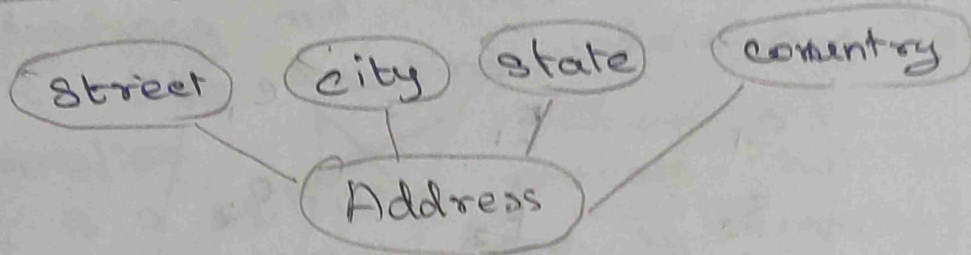
Attribute - defines

the entity type eg: rollno, name, etc...

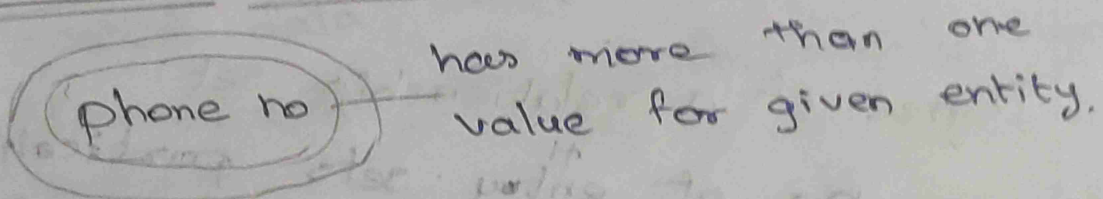
1. Key attribute Rollno.

Uniquely identifies each entity in the entity set is called key attributes.

2. Composite attribute



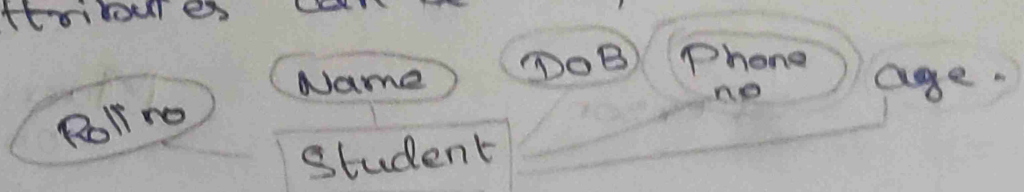
3. Multivalued attribute

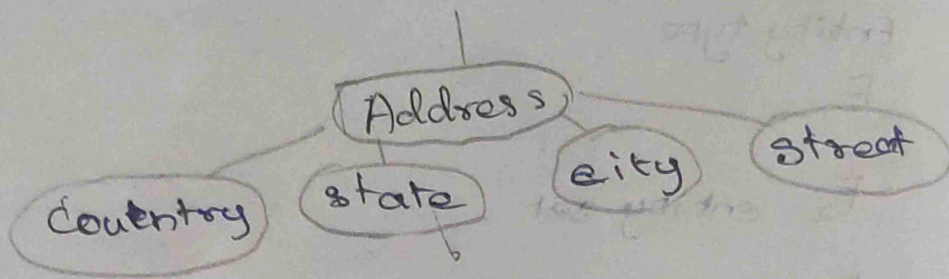


4. Derived attribute

derived from other attributes of the entity types

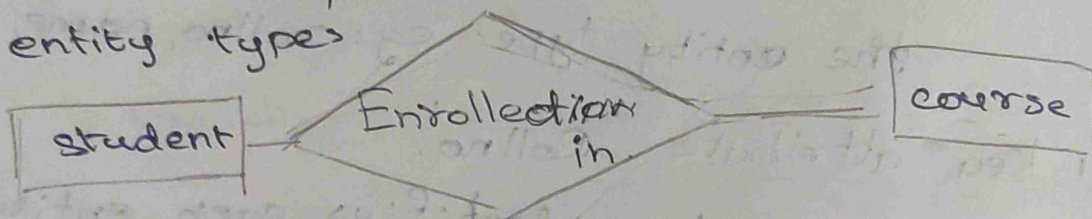
complete entity type student with its attributes can be represented as





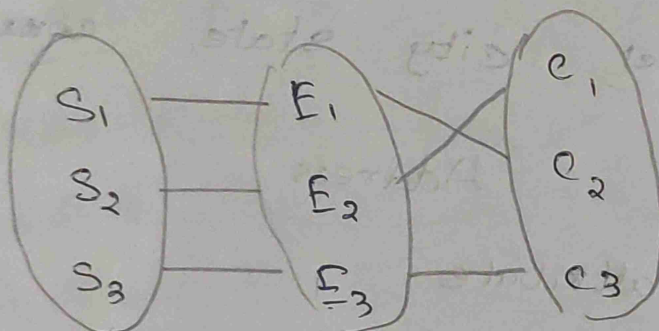
Relationship Type and Relationship Set.

↓
Represents the association between entity types



Relationship set:

set of relationships of some type is known as relationship set.

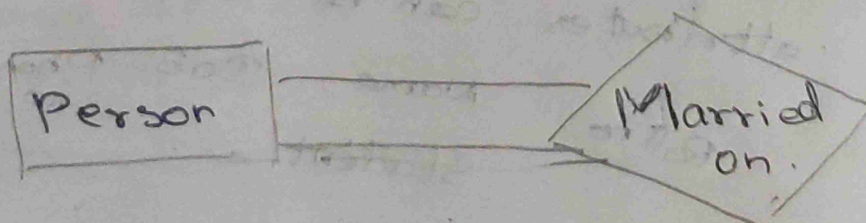


Degree of a relationship set:

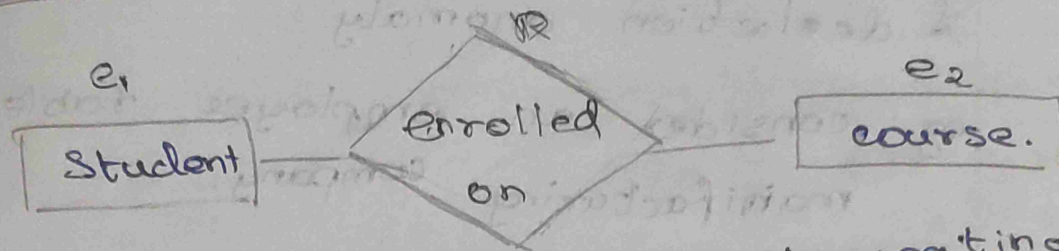
no of entity sets participating in a relationship set is called as degree of a relationship set.

1. Unary Relationship

One entity set participating in a relation



Binary Relationship



Two entities participating in a relation.

13.9.22 Normalization of Database:

- * Database Normalization is a technique of organizing the data in the database.
- * Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like inserting, update and Deletion

Anomalies

- * It is a multistep process that puts data into tabular form by removing duplicated data from the relational tables.
- * Normalization is used for mainly two purpose.

⇒ Eliminating redundant data (useless)

⇒ Ensuring data dependencies make sense (i.e.) data is logically stored

Anomalies → something different from normal (organization)

Anomalies in DBMS

3 types of Anomalies that occur when the database is normalised

These are - ~~the~~ Insertion update & deletion anomaly

eg: consider an employee table of manufacturing company.

Empid	empname	emp-address	emp-dep
101	Adi 1	mdw	Dep 11
101	Adi 1	mdu	Dep 12
202	Aziz	ehemai	Dep 33
203	Karan	Th Trichy	Dep 44

→ Table is not ~~an~~ normalised. If a table is not ~~a~~ normalised, will lead to Anomalies.

Normalization

~~(X)~~ most commonly used ~~a~~ normal form are

(i) First normal form (1NF)

(ii) Second normal form (2NF)

(iii) Third normal form (3NF)

(iv) Boyce & Codd normal form (BCNF)

First normal form (1NF)

* As per 1NF rule, an attribute (Column) of a table cannot hold multiple values.

It should hold only atomic values

"no repeating groups"

eg. a company stores name & address of its employees. It creates a table

empid	empname	emp-address	emp-mobile
101	Adil	FSD	98... 12 73... 41
102	Aziz	LHR	76... 20 80... 10
103	karan	KAR	63... 10

They have more than one value in an attribute emp-mobile.

This is not in 1NF empid 101
X 102 violates the rules.

To make it as in 1NF we have to make the table as follows.

ii) Second normal form (2NF)

Here there must not be any partial dependency of any column on primary key.

(ie) for a table that has concatenated primary key each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence.

* If any column depends only on one part of the concatenated key, then the table fails second normal form.

eg. Student table in 1NF

Student	Age	Subject
Meena	21	Biology
Meena	21	Maths
Raja	22	Maths
Kavin	24	Maths

It satisfies as per 1NF But for 2NF is not in completed form.

⇒ (ie) Age column is dependent only on student, it does not have any dependencies towards subject column.

so it is to be converted to 2NF as follows: split the table as.

Student	Age	Student	Subject
Meena	21	Meena	Biology
Meena	21	Meena	Maths
Raja	22	Raja	Maths
Kavin.	24	Kavin	Maths

eg: 2. 2NF

1NF & all non key attributes are fully dependent on the primary key ("no partial dependencies")

Student	Course-id	Grade	Address
Erik	C15331	A	80, Eriess on
Sven	C15331	B	12, delson street
Ravi	C15331	C	192, del street
Hasen	C15362	A	212, Raja street

Student	Address
Erik	C15331
Sven	C15331
Ravi	C15331
Hasen	C15362

Student	Course id	Grade.
F		

Third Normal Form.

⇒ This applies that every non prime attributes of table must be the case that a non-prime attribute is determined by another non-prime attribute.

⇒ So this transitive functional dependency should be removed from the table and also the table must be in second normal form.

Std.id	std.name	DOB	Street	city	State	zip

In this table student is primary key but street, city & state depends upon zip. The dependency between zip & other fields is called Transitive dependency.

Hence to apply 3NF; we need to move street, city & state to new table with zip as primary key.

New student detail ①

Std-id	std-name	DOB	Zip

② Address table

zip	street	city	state.

The advantage of removing transitive dependency are:

- * Amount of data duplication is reduced
- * Data integrity achieved.