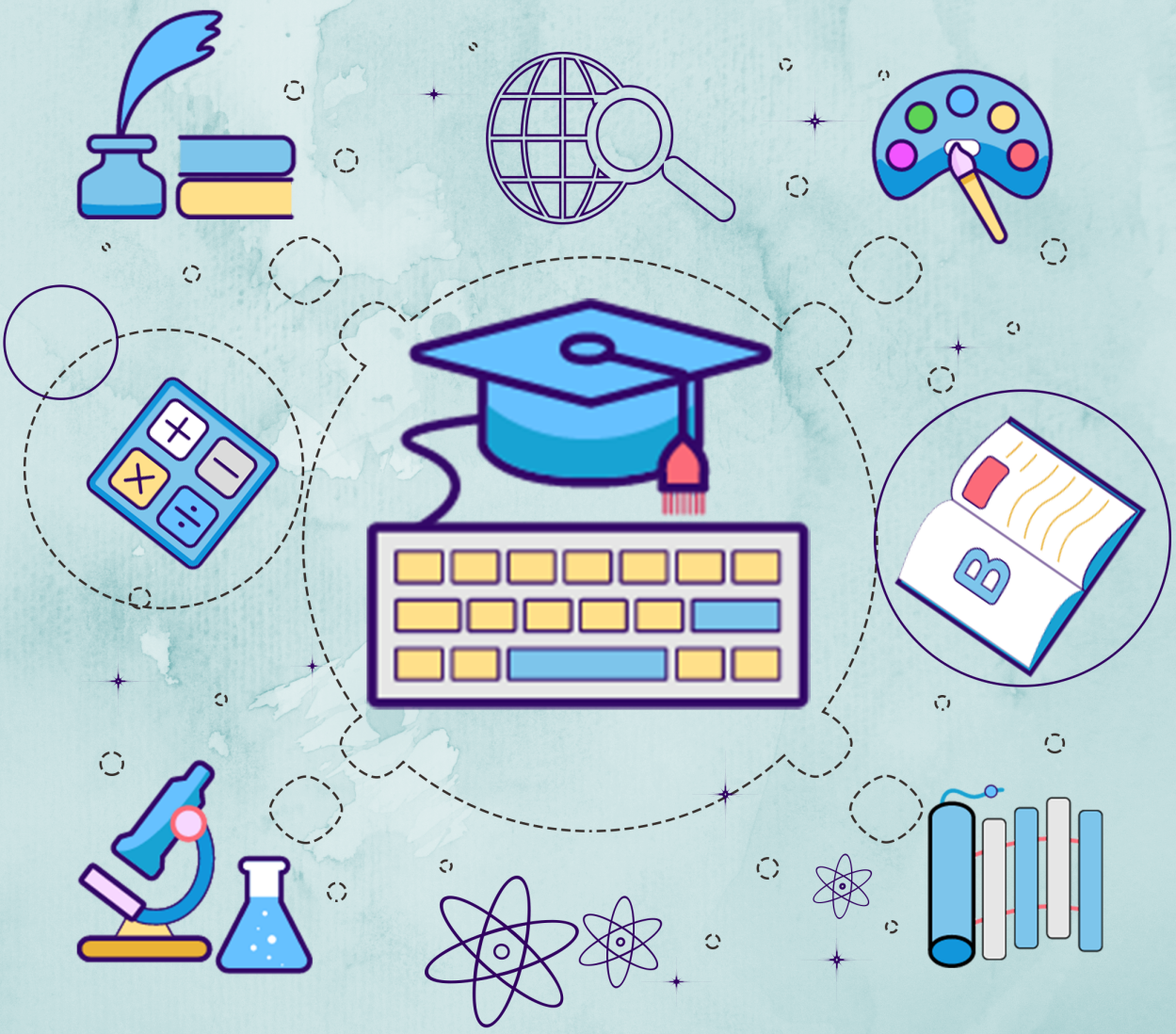
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

# KeralaNotes



SYLLABUS | STUDY MATERIALS | TEXTBOOK

PDF | SOLVED QUESTION PAPERS

www.keralanotes.com

KeralaNotes

# KTU S4 CSE SHORT NOTES

# DATABASE MANAGEMENT SYSTEMS (CST 204)

# Module 2

Related Link :

- KTU S4 CSE NOTES 2019 SCHEME
- KTU S4 SYLLABUS CSE COMPUTER SCIENCE
- KTU PREVIOUS QUESTION BANK S4 CSE SOLVED
- KTU CSE TEXTBOOKS S4 B.TECH PDF DOWNLOAD
- KTU S4 CSE NOTES | SYLLABUS | QBANK | TEXTBOOKS DOWNLOAD

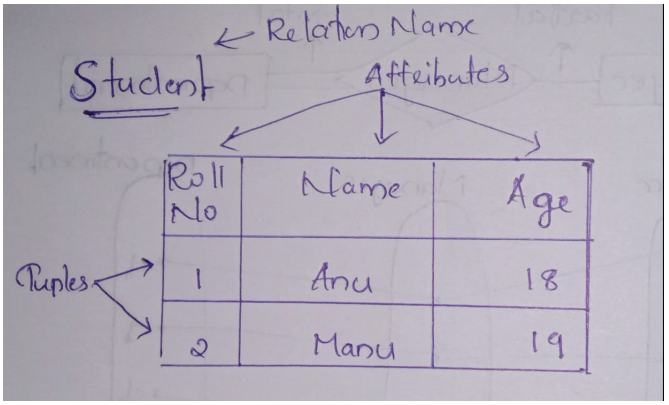
# index





# Relational model

- The relational model represents the database as a collection of relations
- In the formal relational model terminology,
  - ❑A **row** is called a **tuple**
  - ❑A **column** is called an **attribute**
  - ❑The **table** is called a **relation**



## DOMAIN

❑A set of atomic values allowed for an attribute

Eg:Student age

## RELATION SCHEMA

- ❑ Describes a relation
- ❑ Made up of a relation name R and a list of attributes A1,A2,……..An

Eg:STUDENT(Name,Rollnum,Age)

STUDENT(Name:string,Rollnum:integer,Age:integer)

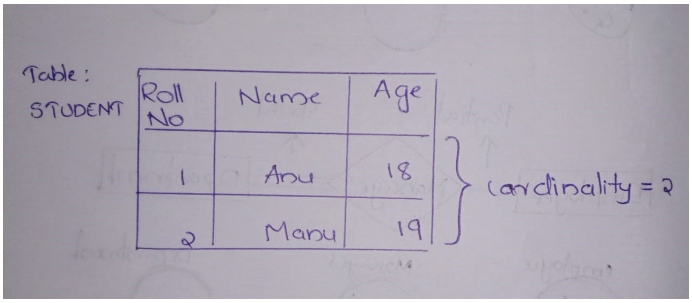
## DEGREE OF A RELATION

❑Number of attributes in a relation schema

Eg: STUDENT(Name,Rollnum,Age)-degree-3

## CARDINALITY

- ❑ Total number of tuples present in a relation



# Relational models constraints

- Constraints are the restrictions or the limitations on data in the database

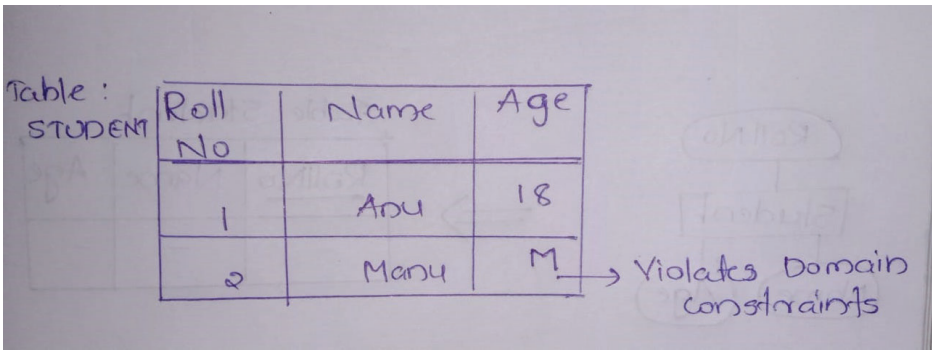
1. **Inherent model-based constraints or implicit constraints:** Constraints that are inherent in the data model
2. **Schema-based constraints or explicit constraints:** Constraints that are defined directly in the schemas of the data model
3. **Application-based or semantic constraints or business rules:** Constraints that cannot be directly expressed in the schemas of the data model, and hence must be expressed and enforced by the application programs.



# Schema based constraints

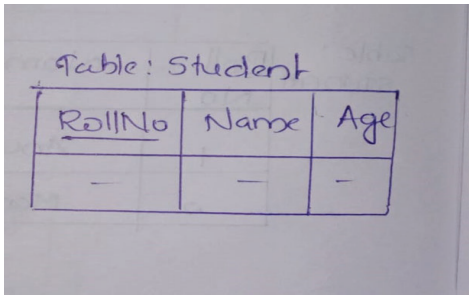
## 1.DOMAIN CONSTRAINTS

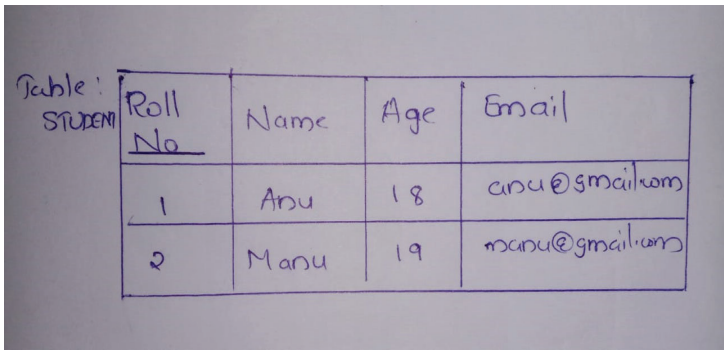
- Must be **atomic value**
- Performs **data type check**



# 2.KEY CONSTRAINTS

- An attribute that can uniquely identify each tuple in a relation is called a **key**

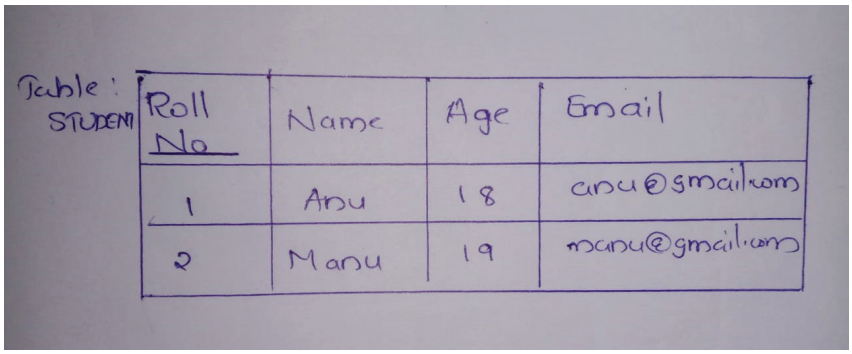



- A **superkey** specifies that no two tuples can have the same value
  - Every relation has atleast one superkey -set of all attributes.EG:{RollNo,Email,{RollNo,Name},.....}



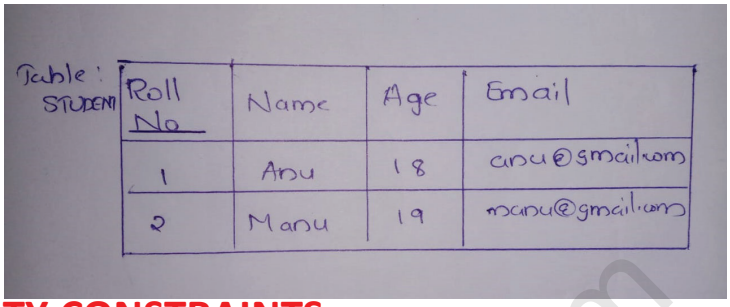
- **Candidate key** is a set of attributes that uniquely identify the tuples in a relation
  Eg:RollNo and Email



## 3.CONSTRAINTS OF NULL VALUES

- Specifies whether null values are permitted or not(**NOT NULL**)

Eg:Name




## 4.ENTITY INTEGRITY CONSTRAINTS

- Specify that **no primary key value can be null**

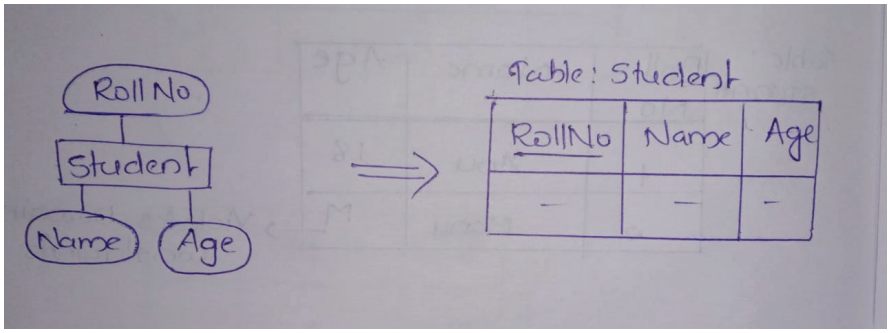


# ER MODEL TO RELATIONAL MODEL

1.Create a table for each entity

2.Entity's attribute should become fields of tables with respect to data types

3.Declare primary key



# Relational Algebra

- It is a **procedural query language**
- It consist of a set of operation that take **one or two relation as input** and produce a **new relation** as their **result**
- It consist of two types of operation

    1.**Unary** operation:They operate on one relation

        Eg:Select,Project

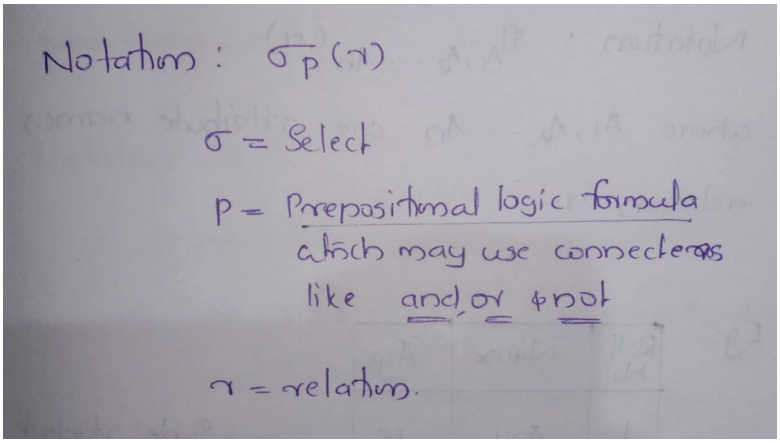
    2.**Binary** operation:They operate on pairs of relationships

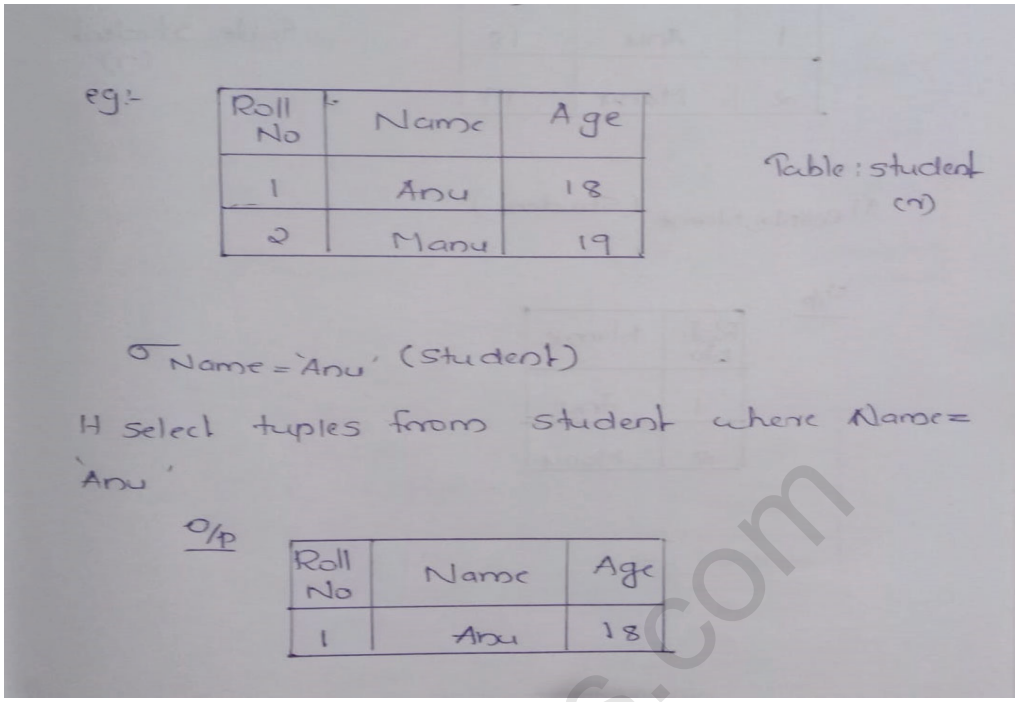
        Eg:Cartesian product



# Select operation

- The select operation **select tuples** that satisfy a given predicate





eg:-

| Roll No | Name | Age |
|---|---|---|
| 1 | Anu | 18 |
| 2 | Manu | 19 |

Table : student (r)

$\sigma_{Name = 'Anu'}(Student)$

H select tuples from student where Name= 'Anu'

O/P

| Roll No | Name | Age |
|---|---|---|
| 1 | Anu | 18 |

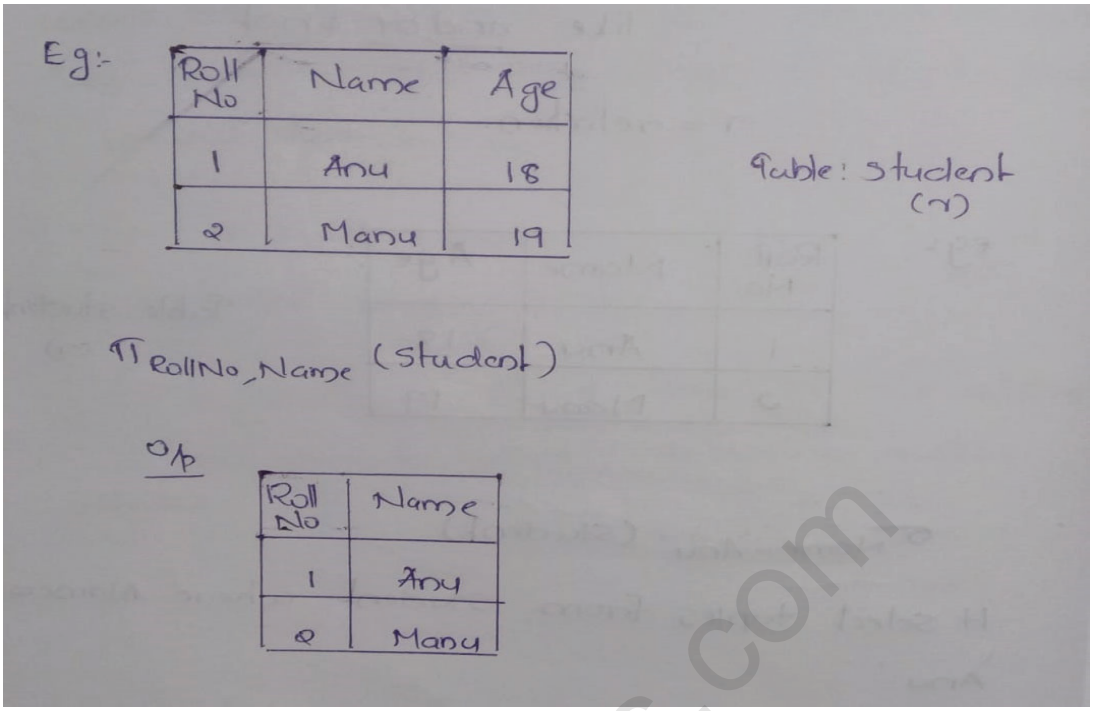


# Project operation

- It **projects attributes(columns)** that satisfy a given predicate

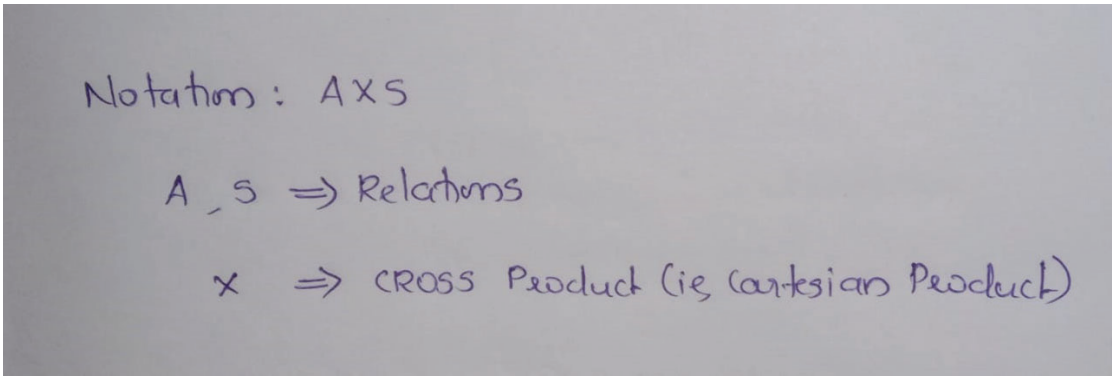
Notation : $\Pi_{A_1, A_2 \ldots A_n}(r)$

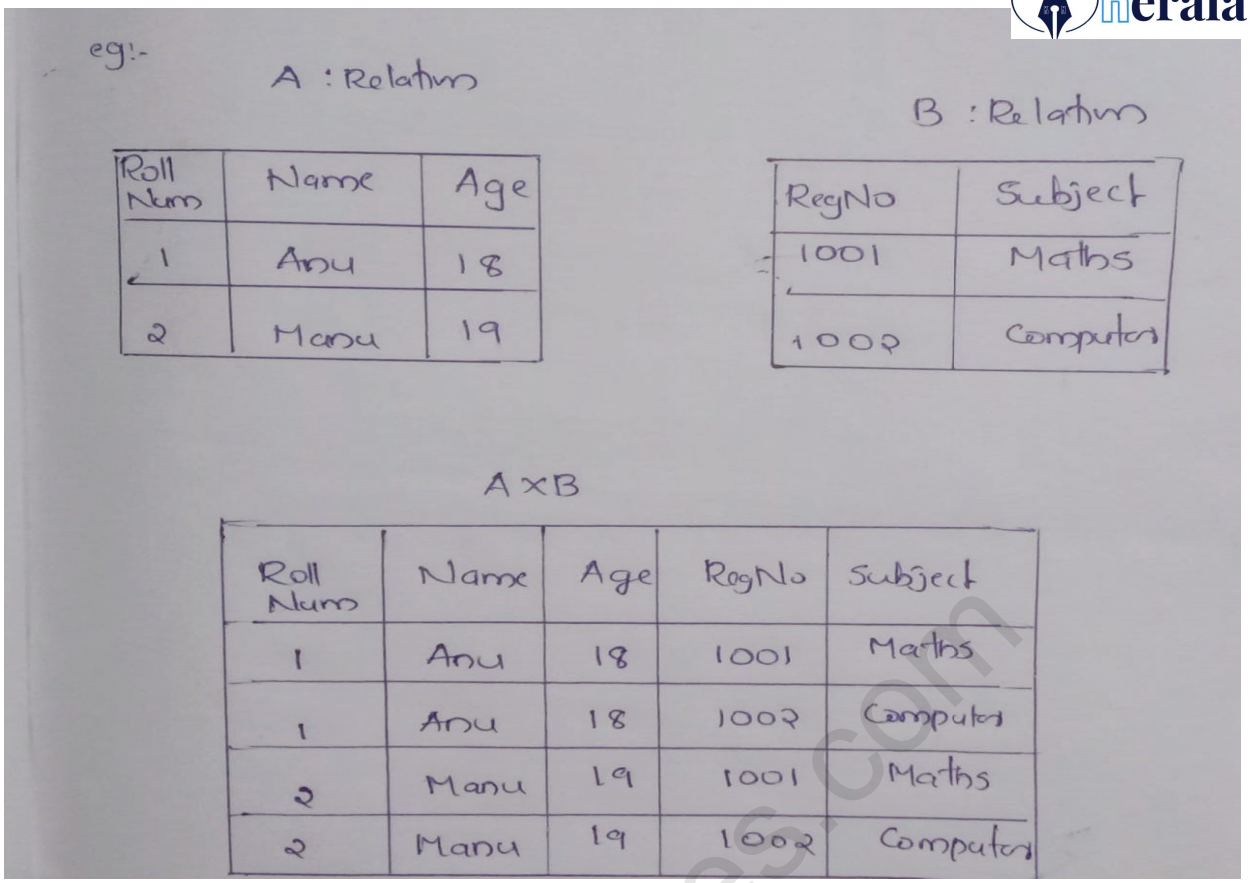
where $A_1, A_2, \ldots A_n$ are attribute names of relation r.




# Cartesian product

- It **combines** information of **two different relations into one relation**





eg:-

A : Relation

| Roll Num | Name | Age |
|---|---|---|
| 1 | Anu | 18 |
| 2 | Manu | 19 |

B : Relation

| RegNo | Subject |
|---|---|
| 1001 | Maths |
| 1002 | Computers |

A x B

| Roll Num | Name | Age | RegNo | Subject |
|---|---|---|---|---|
| 1 | Anu | 18 | 1001 | Maths |
| 1 | Anu | 18 | 1002 | Computers |
| 2 | Manu | 19 | 1001 | Maths |
| 2 | Manu | 19 | 1002 | Computers |



# index

- ❑Structured Query Language (SQL)
- ❑DDL
  1. CREATE
  2. DROP
  3. ALTER
- ❑DML
  1. INSERT
  2. DELETE
  3. UPDATE
- ❑JOIN
  1. EQUI-JOIN
  2. NATURAL JOIN

# sql

- The **SQL**(Structured Query Language) is a standard language **for storing and managing data in a RDBMS**.
- It enables a user to **create,read,update and delete relational databases** and tables
- SQL is **not case sensitive**. Generally, keywords of sql are written in upper case
- SQL comprises DDL and DML. Using DDL, one can design and modify database schema whereas DML used to store and retrieve data from database



# ddl

**CREATE**

**DROP**

**ALTER**

# CREATE

• Creates new **databases,tables and views** from RDBMS

CREATE TABLE tablename(col1 datatype,col2 datatype,….);

Eg: CREATE TABLE student(id int primary key,name varchar(20));



# alter

• **Modifies database schema**(ie, structure of the DB)

ALTER TABLE tablename ADD columnname datatype;

Eg: ALTER TABLE student ADD mark int

# drop

- It is used to **delete whole database or** just a **table**

DROP TABLE|DATABASE tablename|databasename;

Eg: DROP TABLE student;



# DML

**INSERT**

**UPDATE**

**DELETE**

# INSERT

• It is used to **insert data into table**

INSERT INTO tablename(col1,col2,…) VALUES(value1,value2,..);

Eg: INSERT INTO student VALUES(1,ashi);



# update

• It is used to **update existing data** within a table

UPDATE table_name SET col1=val1,col2=val2,.. WHERE condition;

Eg:UPDATE student SET name='anu' WHERE id=1

# delete

- It is used to **delete records from a table**

DELETE FROM tablename WHERE condition;

Eg:DELETE FROM student WHERE id=1;



# join

It is used to **combine two tables based on** a specified **common field** between them

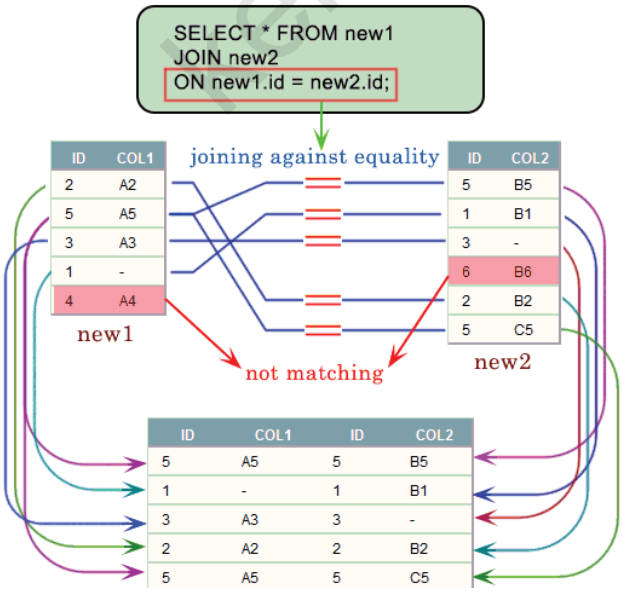
**EQUI-JOIN**

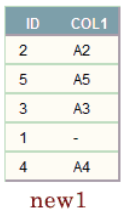
**NATURAL JOIN**

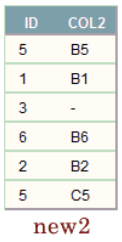
# Equi-join

- It performs a **JOIN against equality or matching column(s) values of the associated tables**
- An equal sign(=) is used as comparison operator in the where clause to refer equality
- SELECT columnlist FROM table1,table2 WHERE table1.columnname = table2.columnname;
- You may also perform EQUI-JOIN by using **JOIN keyword followed by ON keyword**
- SELECT * FROM table1 JOIN table2[ON (join_condition)];





# Natural join

- It is a type of EQUI-JOIN and is structured in such a way that, **columns with the same name of associated tables will appear once only**
- SELECT * FROM table1 NATURAL JOIN table2;



| ITEM_ID | ITEM_NAME | ITEM_UNIT | COMPANY_ID |
|---|---|---|---|
| 1 | Chex Mix | Pcs | 16 |
| 6 | Cheez-It | Pcs | 15 |
| 2 | BN Biscuit | Pcs | 15 |
| 3 | Mighty Munch | Pcs | 17 |
| 4 | Pot Rice | Pcs | 15 |
| 5 | Jaffa Cakes | Pcs | 18 |
| 7 | Salt n Shake | Pcs | - |

| COMPANY_ID | COMPANY_NAME | COMPANY_CITY |
|---|---|---|
| 18 | Order All | Boston |
| 15 | Jack Hill Ltd | London |
| 16 | Akas Foods | Delhi |
| 17 | Foodies. | London |
| 19 | sip-n-Bite. | New York |

** Same column came once

| COMPANY_ID | ITEM_ID | ITEM_NAME | ITEM_UNIT | COMPANY_NAME | COMPANY_CITY |
|---|---|---|---|---|---|
| 16 | 1 | Chex Mix | Pcs | Akas Foods | Delhi |
| 15 | 6 | Cheez-It | Pcs | Jack Hill Ltd | London |
| 15 | 2 | BN Biscuit | Pcs | Jack Hill Ltd | London |
| 17 | 3 | Mighty Munch | Pcs | Foodies. | London |
| 15 | 4 | Pot Rice | Pcs | Jack Hill Ltd | London |
| 18 | 5 | Jaffa Cakes | Pcs | Order All | Boston |