JAIN DEEMED-TO-BE UNIVERSITY

SCHOOL OF COMPUTER APPLICATIONS & INFORMATION TECHNOLOGY

Department of Master of Information Technology

# 18MSIT2H04 - Database Management System

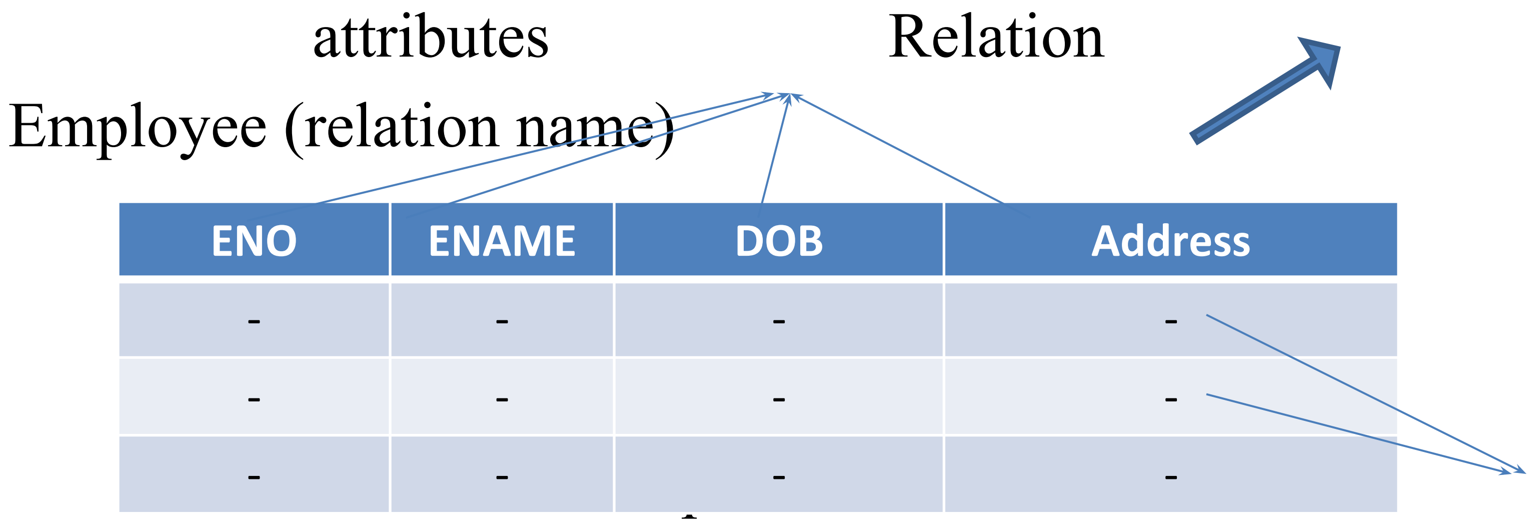
## Unit 2

## Relational Model

Haripriya V Assistant Professor
School of CA & IT

## Introduction

- It is used for data storage and processing.
- A relation is a table with column and rows.
- It is based on the mathematical concept of relation which is represented physically as table.



| ENO | ENAME | DOB | Address |
|---|---|---|---|
| - | - | - | - |
| - | - | - | - |
| - | - | - | - |

## Terms

- Degree of relation - number of attributes in a relation.
  - Ex: in the previous table degree is 4
- Cardinality of relation – number of tuples in a relation.
  - Ex: in the previous table cardinality is 3

| Informal terms | Formal terms |
|---|---|
| Table | Relation |
| Column header / field | Attributes |
| All possible column values | Domain |
| Row | Tuple |
| Table definition | Scheme of relation |

Unit 2
Relational Model

## Properties of relations

- The relation has name that is distinct from all the names in the relationship schema.
- Each cell of the relation contains exactly one value.
- Each attribute has distinct names.
- Each tuple is distinct, there are no duplicate tuples.
- The order of attribute has no significance.
- The order of tuple has no significance.

## Constraints

-  Every relation has some condition that must hold for it to be a valid relation, called Relation Integrity Constraint.

Constraint types

-  Key constraints / entity constraint
-  Domain constraint
-  Referential integrity constraint

# Constraints types

<u>Key constraint:</u>

- In a relation with a key attribute no two tuples can have identical values for key attribute.
- Key attribute cannot have null values.
- If there are more than one such minimal subset, these are called candidate key.

<u>Domain constraint:</u>

- Every attribute is bound to have a specific range of values.
- Ex: age cannot have negative value or less than zero.

  phone number should be between 0-9

<u>Referential integrity</u>:

- Works on concepts of foreign key.
- A foreign key is a key attribute of a relation that can be defined in other relation.

DATABASE
MANAGEMENT
SYSTEM
DBMS

## Unit 2
## Relational Model

# Keys

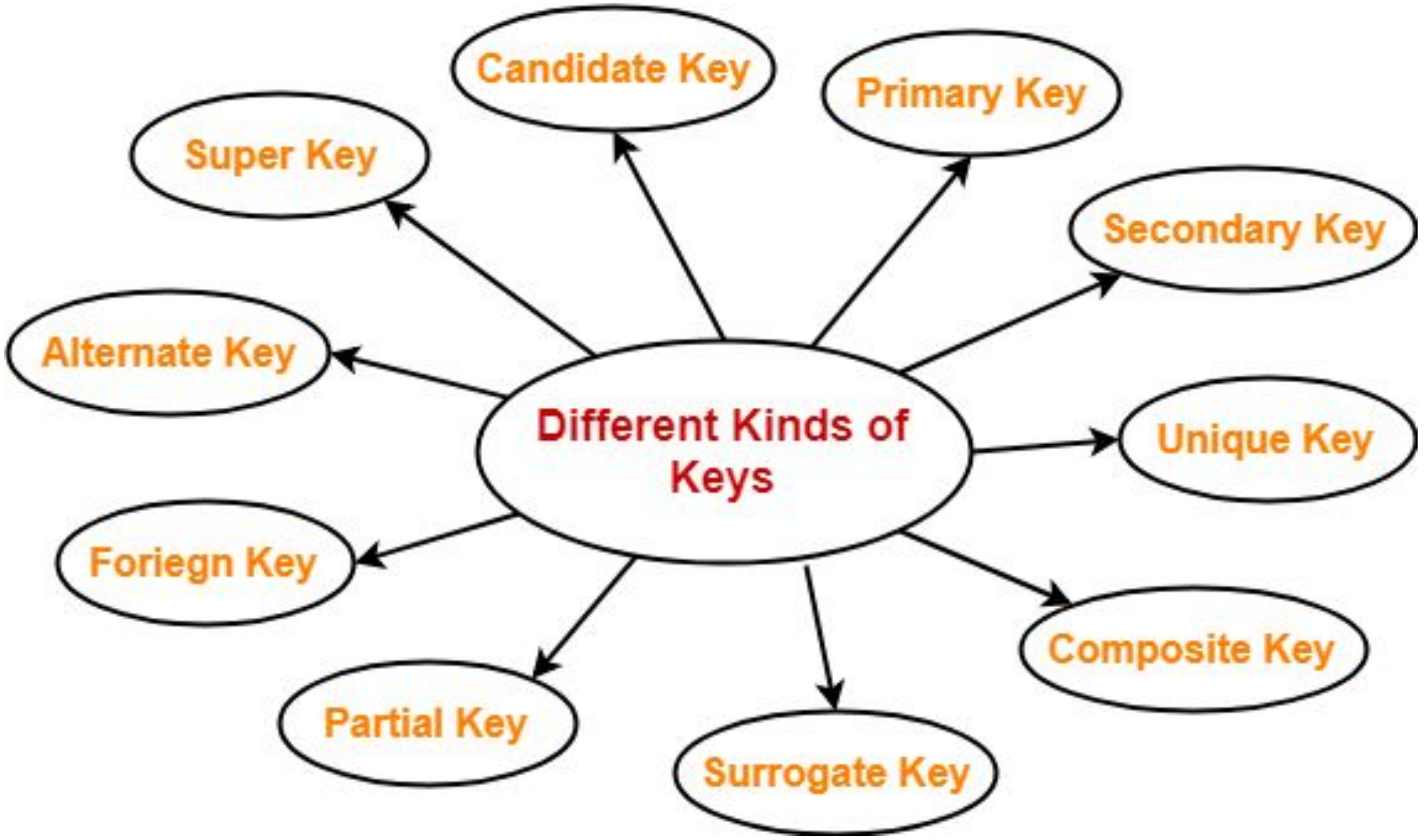
<u>What is the use of keys?</u>

- Keys are used to establish and identify relation between tables.
- Each record within a table can be uniquely identified by combination of one or more fields in a tables.
- It help to enforce integrity and identify relationship.

<u>Why we need key?</u>

- In a real application huge data has to store in tables.
- Tables extends to thousands of records stored in, unsorted/unorganized.
- To fetch any record.
- To avoid all these keys are define to easily identify any new row of data in a table.

# Types of keys



## Super key

- A super key is a set of attributes that can identify each tuple uniquely in the given relation.
- A super key is not restricted to have any specific number of attributes.
- Thus, a super key may consist of any number of attributes.

- **<u>Example-</u>**
- Consider the following Student schema-
- **Student ( roll , name , sex , age , address , class , section )**

- Given below are the examples of super keys since each set can uniquely identify each student in the Student table-
- ( roll , name , sex , age , address , class , section )
- ( class , section , roll )
- (class , section , roll , sex )
- ( name , address )

## Primary key

- A primary key is a candidate key that the database designer selects while designing the database.

**OR**

- Candidate key that the database designer implements is called as a primary key.

**NOTES-**

- The value of primary key can never be NULL.
- The value of primary key must always be unique.
- The values of primary key can never be changed i.e. no updation is possible.
- The value of primary key must be assigned when inserting a record.

**Secondary Key:**

- It is required for the indexing purpose for better and faster searching.

Unit 2
Relational Model

## Alternate key

- All the keys which are not primary key are called an alternate key.
- It is a candidate key which is currently not the primary key.
- However, A table may have single or multiple choices for the primary key.

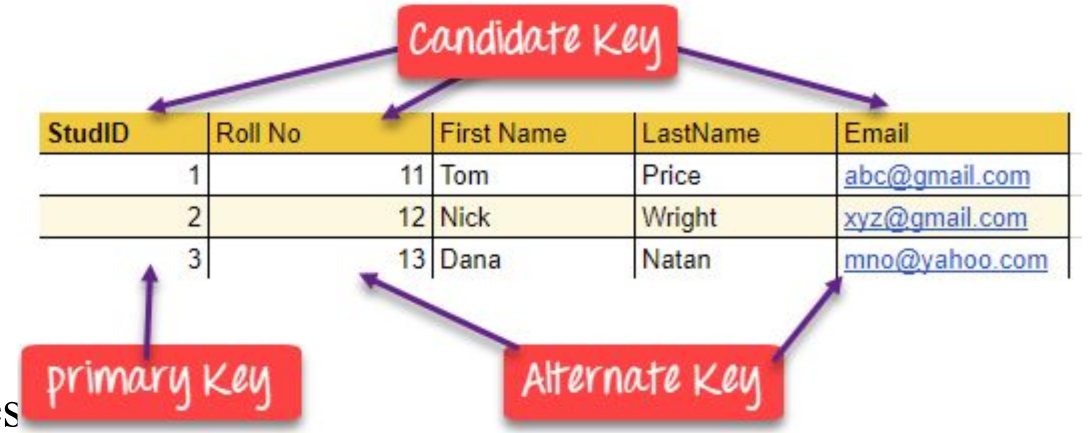
- Example: In this table.

StudID, Roll No, Email are qualified to become a primary key. But since StudID is the primary key, Roll No, Email becomes the alternative key.

Unit 2
Relational Model

# Candidate key

- A super key with no repeated attribute is called candidate key.
- The Primary key should be selected from the candidate keys. Every table must have at least a single candidate key.



| StudID | Roll No | First Name | LastName | Email |
|---|---|---|---|---|
| 1 | 11 | Tom | Price | abc@gmail.com |
| 2 | 12 | Nick | Wright | xyz@gmail.com |
| 3 | 13 | Dana | Natan | mno@yahoo.com |

**Properties of Candidate key:**

- It must contain unique values
- Candidate key may have multiple attributes
- Must not contain null values
- It should contain minimum fields to ensure uniqueness
- Uniquely identify each record in a table.

## Unique key

- Unique key is a key with the following properties-
- It is unique for all the records of the table.
- Once assigned, its value can not be changed i.e. it is non-updatable.
- It may have a NULL value.

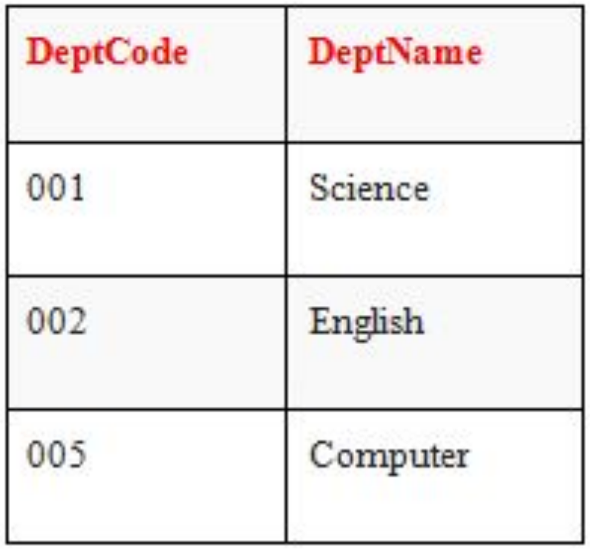
**Example-**

- 
- The best example of unique key is **Adhaar Card Numbers**.
- The Adhaar Card Number is unique for all the citizens (tuples) of India (table).
- If it gets lost and another duplicate copy is issued, then the duplicate copy always has the same number as before.
- Thus, it is non-updatable.
- Few citizens may not have got their Adhaar cards, so for them its value is NULL.

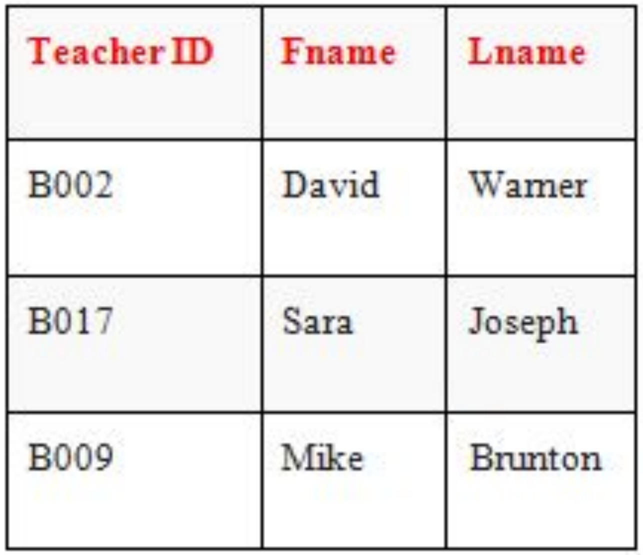
Unit 2
Relational Model

## Foreign key

- A foreign key is a column which is added to create a relationship with another table.
- Foreign keys help us to maintain data integrity and also allows navigation between two different instances of an entity.
- Every relationship in the model needs to be supported by a foreign key.



| DeptCode | DeptName |
|---|---|
| 001 | Science |
| 002 | English |
| 005 | Computer |



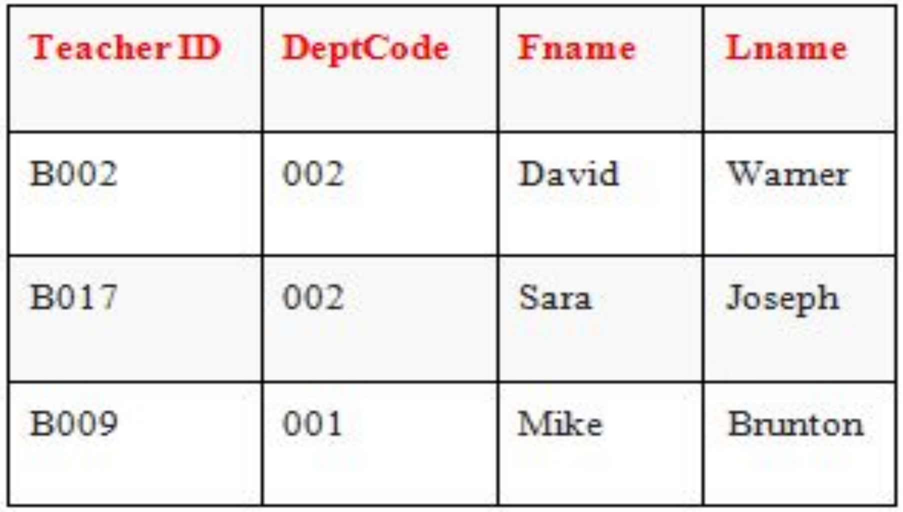
| Teacher ID | Fname | Lname |
|---|---|---|
| B002 | David | Wamer |
| B017 | Sara | Joseph |
| B009 | Mike | Brunton |

DATABASE
MANAGEMENT
SYSTEM
DBMS

Unit 2
Relational Model

## Foreign key

- In this example, we have two table, teach and department in a school. However, there is no way to see which search work in which department.
- In this table, adding the foreign key in Deptcode to the Teacher name, we can create a relationship between the two tables



| Teacher ID | DeptCode | Fname | Lname |
|---|---|---|---|
| B002 | 002 | David | Warner |
| B017 | 002 | Sara | Joseph |
| B009 | 001 | Mike | Brunton |

This concept is also known as Referential Integrity.

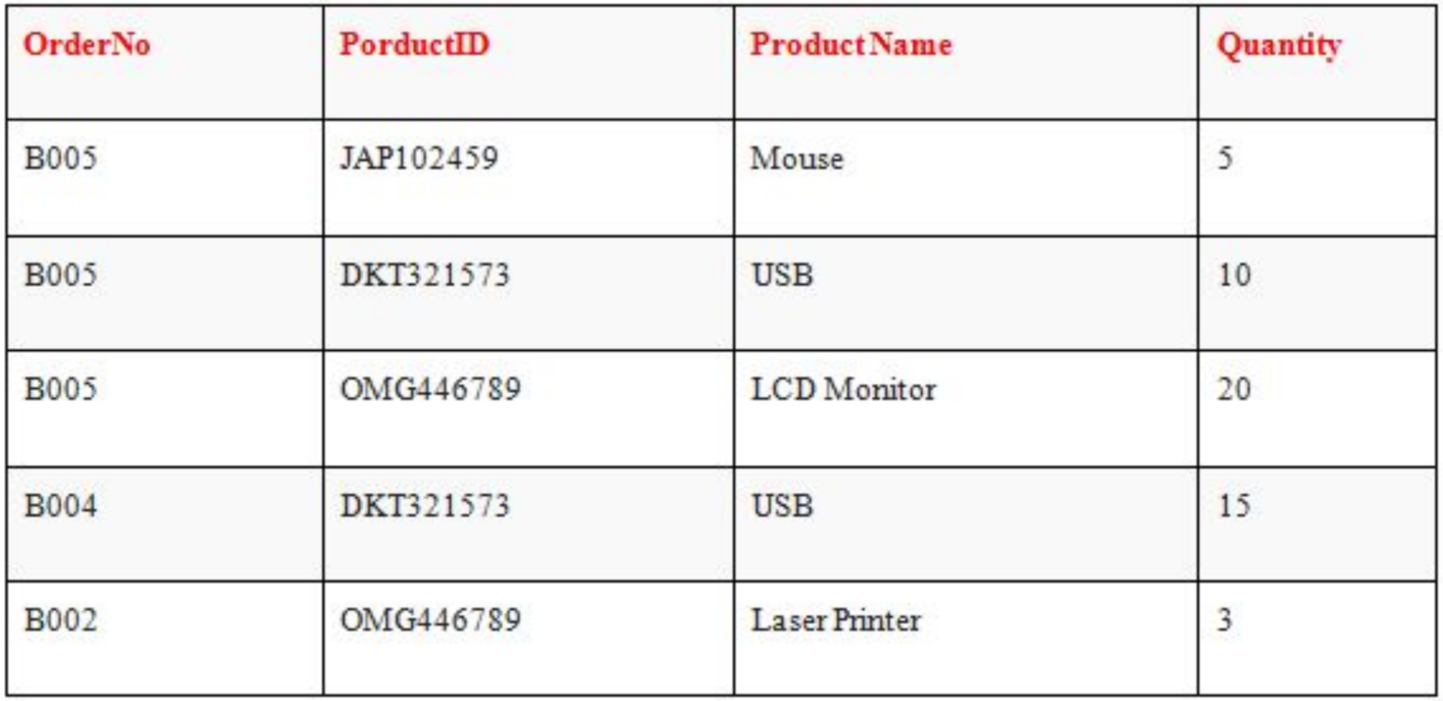
## Compound key

- Compound key has many fields which allow you to uniquely recognize a specific record.
- It is possible that each column may be not unique by itself within the database.
- However, when combined with the other column or columns the combination of composite keys become unique.

## Unit 2
## Relational Model

# Compound key

In this example, OrderNo and ProductID can't be a primary key as it does not uniquely identify a record. However, a compound key of Order ID and Product ID could be used as it uniquely identified each record.



| OrderNo | PorductID | Product Name | Quantity |
|---|---|---|---|
| B005 | JAP102459 | Mouse | 5 |
| B005 | DKT321573 | USB | 10 |
| B005 | OMG446789 | LCD Monitor | 20 |
| B004 | DKT321573 | USB | 15 |
| B002 | OMG446789 | Laser Printer | 3 |

## Partial key

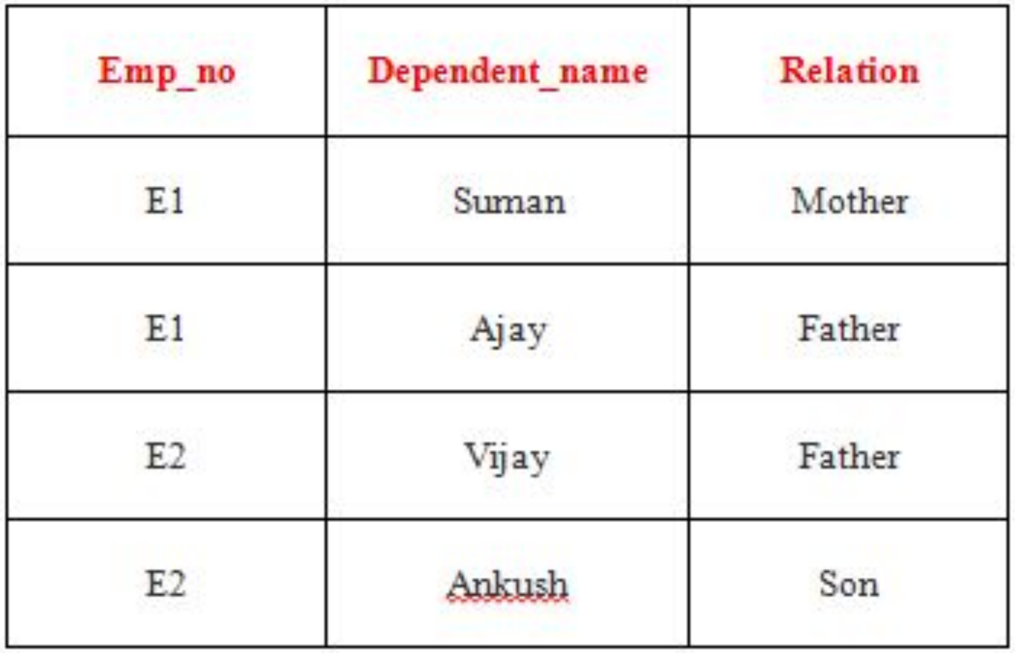
- Partial key is a key using which all the records of the table can not be identified uniquely.
- However, a bunch of related tuples can be selected from the table using the partial key.
- **<u>Example-</u>**

Consider the following schema-

**Department ( <u>Emp_no</u> , Dependent_name , Relation )**

## Partial key

- Here, using partial key Emp_no, we can not identify a tuple uniquely but we can select a bunch of tuples from the table.



| Emp_no | Dependent_name | Relation |
|---|---|---|
| E1 | Suman | Mother |
| E1 | Ajay | Father |
| E2 | Vijay | Father |
| E2 | Ankush | Son |

## Surrogate key

**Surrogate Key:**

Surrogate key is a key with the following properties-

It is unique for all the records of the table.

It is updatable.

It can not be NULL i.e. it must have some value.

**Example-**

 Mobile Number of students in a class, where every student owns a mobile phone.

DATABASE MANAGEMENT SYSTEM
DBMS

# Unit 2
# Relational Model

## Difference between Primary and Foreign key

| Primary Key | Foreign Key |
|---|---|
| Helps you to uniquely identify a record in the table. | It is a field in the table that is the primary key of another table. |
| Primary Key never accept null values. | A foreign key may accept multiple null values. |
| Primary key is a clustered index and data in the DBMS table are physically organized in the sequence of the clustered index. | A foreign key cannot automatically create an index, clustered or non-clustered. However, you can manually create an index on the foreign key. |
| You can have the single Primary key in a table. | You can have multiple foreign keys in a table. |

## Relational Algebra

- Relational algebra is a widely used procedural query language.
- It collects instances of relations as input and gives occurrences of relations as output.
- It uses various operation to perform this action.
- Relational algebra operations are performed recursively on a relation.
- The output of these operations is a new relation, which might be formed from one or more input relations.

## Relational Algebra Operations

**Unary Relational Operations (Fundamental Operations)**

- SELECT (symbol: σ)
- PROJECT (symbol: π)
- RENAME (symbol: ρ )

**Relational Algebra Operations From Set Theory (Additional Operations)**

- UNION (υ)
- INTERSECTION (∩),
- DIFFERENCE (-)
- CARTESIAN PRODUCT ( x )

**Binary Relational Operations**

- JOIN
- DIVISION

Unit 2
Relational Model

## SELECT (σ)

- The SELECT operation is used for selecting a subset of the tuples according to a given selection condition.
- Sigma(σ)Symbol denotes it.
- It is used as an expression to choose tuples which meet the selection condition.
- Select operation selects tuples that satisfy a given predicate.

$$\sigma_p(r)$$

- σ is the predicate
- r stands for relation which is the name of the table
- p is prepositional logic

## SELECT (σ)

- **Example 1**

$\sigma_{\text{topic = "Database"}}\text{(Tutorials)}$

**Output** - Selects tuples from Tutorials where topic = 'Database'.

- **Example 2**

$\sigma_{\text{topic = "Database" and author = “Ramakrishna"}}\text{( Tutorials)}$

**Output** - Selects tuples from Tutorials where the topic is 'Database' and 'author' is Ramakrishna.

- **Example 3**

$\sigma_{\text{sales} > 50000}\text{(Customers)}$

**Output** - Selects tuples from Customers where sales is greater than 50000

## Projection(π)

- The projection eliminates all attributes of the input relation but those mentioned in the projection list.
- The projection method defines a relation that contains a vertical subset of Relation.
- This helps to extract the values of specified attributes to eliminates duplicate values.
- (pi) The symbol used to choose attributes from a relation.
- This operation helps you to keep specific columns from a relation and discards the other columns.

## Projection(π)

**Example of Projection:**
**Customers**

| CustomerID | CustomerName | Status |
|---|---|---|
| 1 | Google | Active |
| 2 | Amazon | Active |
| 3 | Apple | Inactive |
| 4 | Alibaba | Active |

The projection of CustomerName and status will give

$\Pi_{\text{CustomerName, Status}}(\text{Customers})$

| CustomerName | Status |
|---|---|
| Google | Active |
| Amazon | Active |
| Apple | Inactive |
| Alibaba | Active |

## Rename (ρ)

- The rename operation is used to rename the output relation.
- It is denoted by **rho** (ρ).

**ρ(Relation2, Relation1)**

- **Example:**
- We can use the rename operator to rename STUDENT relation to STUDENT1.

ρ(STUDENT1, STUDENT)

- If you want to create a relation STUDENT_NAMES with ROLL_NO and NAME from STUDENT, it can be done using rename operator as:

$$\boldsymbol{\rho(\text{STUDENT\_NAMES}, \prod_{(\text{ROLL\_NO, NAME})}(\text{STUDENT}))}$$

Unit 2
Relational Model

## Union operation (υ)

- UNION is symbolized by ∪ symbol.
- It includes all tuples that are in tables A or in B.
- It also eliminates duplicate tuples.
- So, set A UNION set B would be expressed as:

  The result <- A ∪ B
- For a union operation to be valid, the following conditions must hold -
- A and B must be the same number of attributes.
- Attribute domains need to be compatible.
- Duplicate tuples should be automatically removed.

DATABASE MANAGEMENT SYSTEM
DBMS

Unit 2
Relational Model

## Union operation (υ)

| Table A | |
|---|---|
| column 1 | column 2 |
| 1 | 1 |
| 1 | 2 |

| Table B | |
|---|---|
| column 1 | column 2 |
| 1 | 1 |
| 1 | 3 |

A ∪ B gives →

| Table A ∪ B | |
|---|---|
| column 1 | column 2 |
| 1 | 1 |
| 1 | 2 |
| 1 | 3 |

# Set Difference (-)

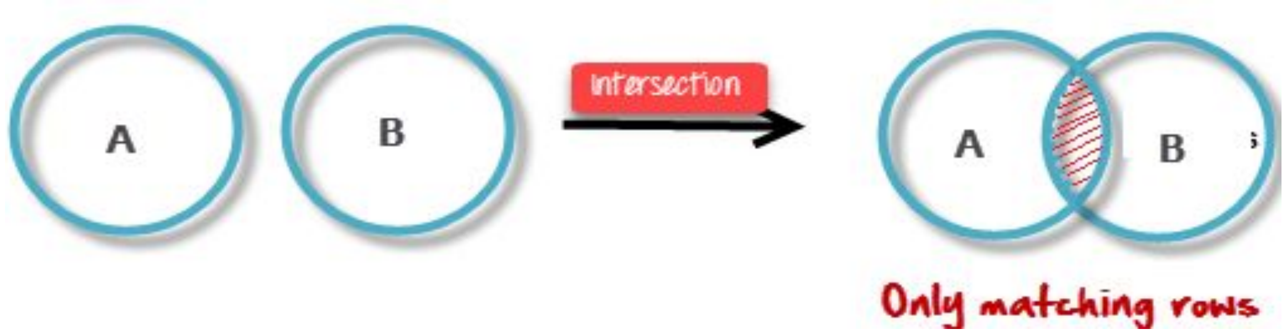
- - Symbol denotes it.
- The result of A - B, is a relation which includes all tuples that are in A but not in B.
- The attribute name of A has to match with the attribute name in B.
- The two-operand relations A and B should be either compatible or Union compatible.
- It should be defined relation consisting of the tuples that are in relation A, but not in B.

| Table A – B | |
|---|---|
| column 1 | column 2 |
| 1 | 2 |

# Intersection (∩)

- An intersection is defined by the symbol ∩

  A ∩ B
- Defines a relation consisting of a set of all tuple that are in both A and B. However, A and B must be union-compatible.



| Table A ∩ B | |
|---|---|
| column 1 | column 2 |
| 1 | 1 |

# Cartesian product(X)

- This type of operation is helpful to merge columns from two relations.
- A Cartesian product is never a meaningful operation when it performs alone.
- However, it becomes meaningful when it is followed by other operations.
- **Example – Cartesian product**

$\sigma_{\text{column 2} = \text{'1'}} (A \text{ X } B)$

- Output – The above example shows all rows from relation A and B whose column 2 has value 1

| **σ column 2 = '1' (A X B)** | |
|---|---|
| **column 1** | column 2 |
| **1** | 1 |
| **1** | 1 |

DATABASE MANAGEMENT SYSTEM

DBMS

Unit 2
Relational Model

# Join Operations

- Join operation is essentially a cartesian product followed by a selection criterion.
- Join operation denoted by $\bowtie$.
- JOIN operation also allows joining variously related tuples from different relations.

**Types of JOIN:**

| **Inner Joins:** | **Outer join:** |
|---|---|
| Theta join | Left Outer Join |
| EQUI join | Right Outer Join |
| Natural join | Full Outer Join |

DATABASE MANAGEMENT SYSTEM

DBMS

Unit 2
Relational Model

## Joins

**Inner Join**

- In an inner join, only those tuples that satisfy the matching criteria are included, while the rest are excluded.
- Let's study various types of Inner Joins:

**Theta Join:**

- The general case of JOIN operation is called a Theta join.
- It is denoted by symbol **θ**

Example

$A \bowtie_{\theta} B$

- Theta join can use any conditions in the selection criteria.
- For example:

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

| A ⋈ A.column 2 > B.column 2 (B) | |
|---|---|
| column 1 | column 2 |
| 1 | 2 |

## Joins

**EQUI join:**

- When a theta join uses only equivalence condition, it becomes a equi join.
- For example:

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

| A ⋈ A.column 2 = B.column 2 (B) | |
|---|---|
| **column 1** | column 2 |
| **1** | 1 |

- EQUI join is the most difficult operations to implement efficiently in an RDBMS

## Joins

### NATURAL JOIN (⋈)

- Natural join can only be performed if there is a common attribute (column) between the relations.
- The name and type of the attribute must be same.

**C**

| Num | Square |
|---|---|
| 2 | 4 |
| 3 | 9 |

**D**

| Num | Cube |
|---|---|
| 2 | 8 |
| 3 | 18 |

**C ⋈ D**

| Num | Square | Cube |
|---|---|---|
| 2 | 4 | 8 |
| 3 | 9 | 18 |

## Joins

### OUTER JOIN

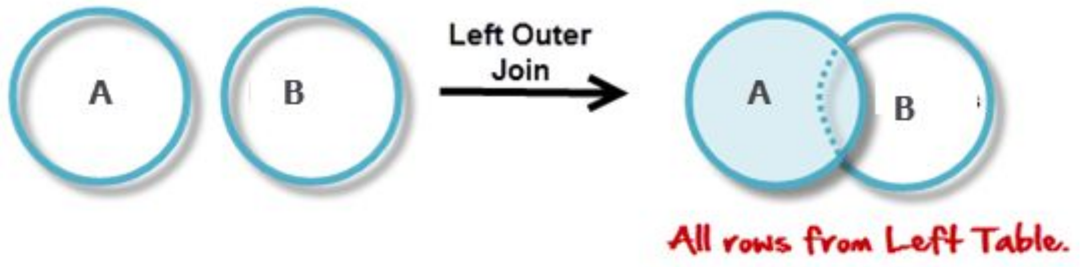
- In an outer join, along with tuples that satisfy the matching criteria, we also include some or all tuples that do not match the criteria.

### Left Outer Join(A⟕ B)

- In the left outer join, operation allows keeping all tuple in the left relation.
- However, if there is no matching tuple is found in right relation, then the attributes of right relation in the join result are filled with null values.

# Unit 2
# Relational Model

## Joins



| A | |
|---|---|
| **Num** | **Square** |
| **2** | 4 |
| **3** | 9 |
| **4** | 16 |

| B | |
|---|---|
| **Num** | **Cube** |
| **2** | 8 |
| **3** | 18 |
| **5** | 75 |

$A ⟕ B$

| A ⋈ B | | |
|---|---|---|
| **Num** | **Square** | **Cube** |
| **2** | 4 | 4 |
| **3** | 9 | 9 |
| **4** | 16 | - |

Unit 2
Relational Model

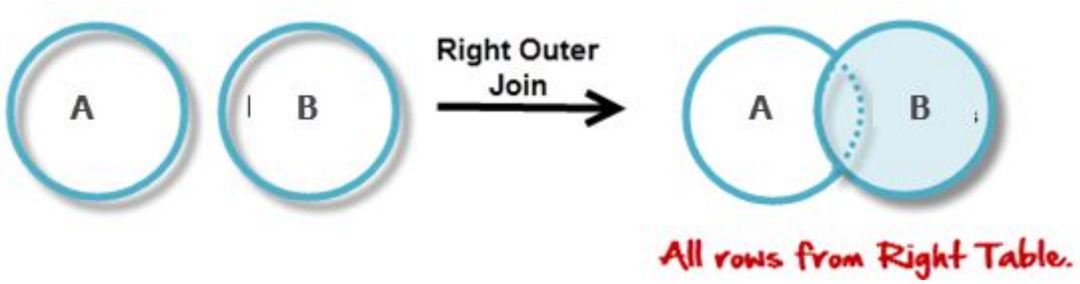
## Joins

### Right Outer Join: ( A ⟖ B )

- In the right outer join, operation allows keeping all tuple in the right relation.
- However, if there is no matching tuple is found in the left relation, then the attributes of the left relation in the join result are filled with null values.

A ⟖ B

| A ⋈ B | | |
|---|---|---|
| **Num** | Cube | Square |
| **2** | 8 | 4 |
| **3** | 18 | 9 |
| **5** | 75 | - |



## Joins

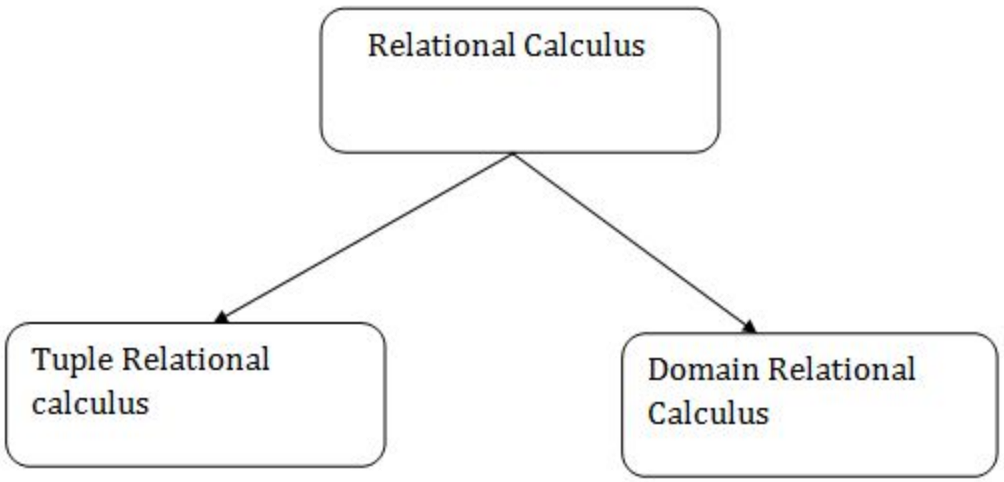
**Full Outer Join: ( A ⟗ B)**

- In a full outer join, all tuples from both relations are included in the result, irrespective of the matching condition.
- A ⟗

| A ⋈ B | | |
|---|---|---|
| **Num** | Cube | Square |
| **2** | 4 | 8 |
| **3** | 9 | 18 |
| **4** | 16 | - |
| **5** | - | 75 |

# Relational Calculus

- Relational calculus is a non-procedural query language.
- In the non-procedural query language, the user is concerned with the details of how to obtain the end results.
- The relational calculus tells what to do but never explains how to do.



## Tuple Relational Calculus (TRC)

- The tuple relational calculus is specified to select the tuples in a relation.
- Tuple Relational Calculus in DBMS uses a tuple variable (t) that goes to each row of the table and checks if the predicate is true or false for the given row. Depending on the given predicate condition, it returns the row or part of the row.

**Notation:**

{T | P (T)}   or {T | Condition (T)}

Where

**T** is the resulting tuples

**P(T)** is the condition used to fetch T. or predicate logic

Unit 2
Relational Model

# Tuple Relational Calculus (TRC)

-  A non procedural query language in the form of {t | p(t)}.
-  It is a set of all tuples t such that predicate p is true for t.
-  T is tuple variable and t[A] denotes the value of tuple t on attribute A.
-  t ∈ s denotes that tuple t is in relation s.
-  P is a formula similar to that of predicate calculus.
-  Predicates calculus formula contains
-  Attribute and constants

# Tuple Relational Calculus (TRC)

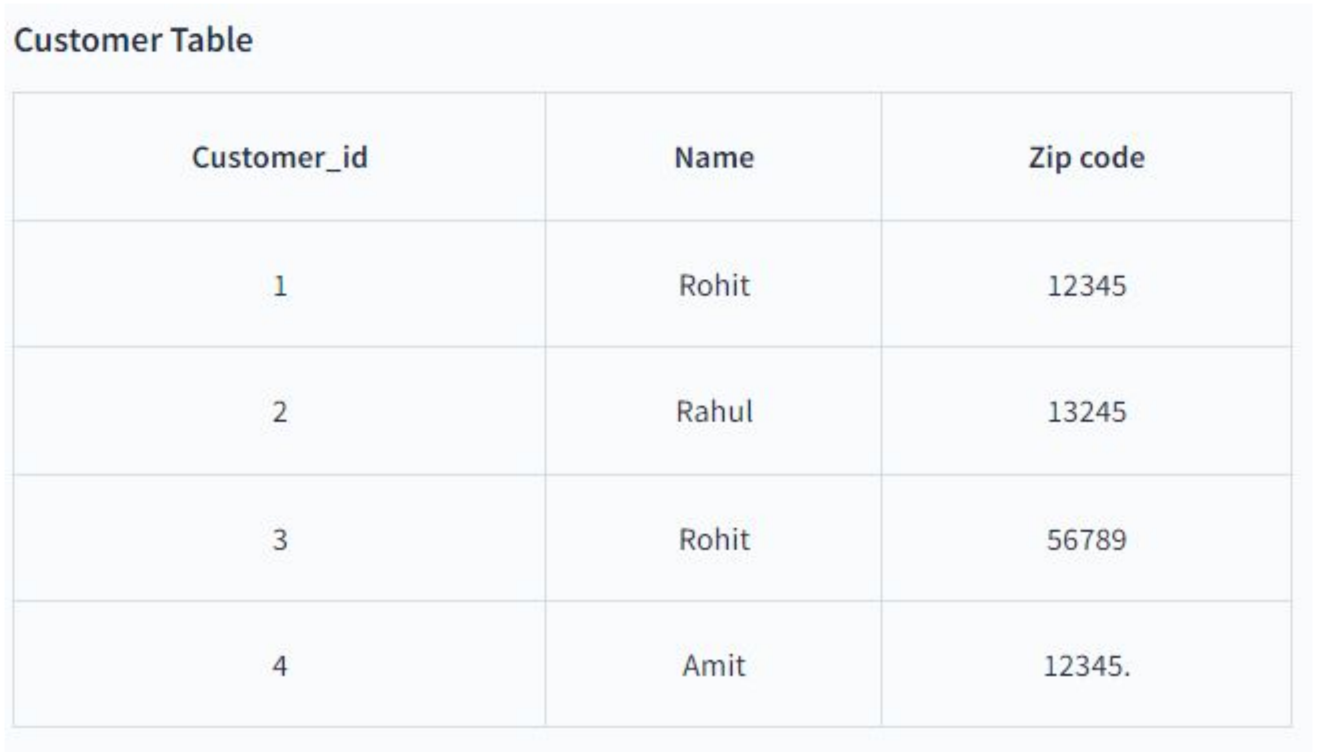
-  Comparison operators <, >, =, <=, >=, !=
-  Connectives $\wedge$ (and), $\vee$ (or), ¬ (not)
-  Implications (=>)

  X => y, if x is true, then y is true
-  Quantifies:

$\exists$ t $\in$ r(q(t)) = “there exists” a tuple in r such that q(t) is true.

$\forall$ t $\in$ r(q(t)) = q is true “for all” tuples t in relation r.

## TRC - Example



Customer Table

| Customer_id | Name | Zip code |
|---|---|---|
| 1 | Rohit | 12345 |
| 2 | Rahul | 13245 |
| 3 | Rohit | 56789 |
| 4 | Amit | 12345. |

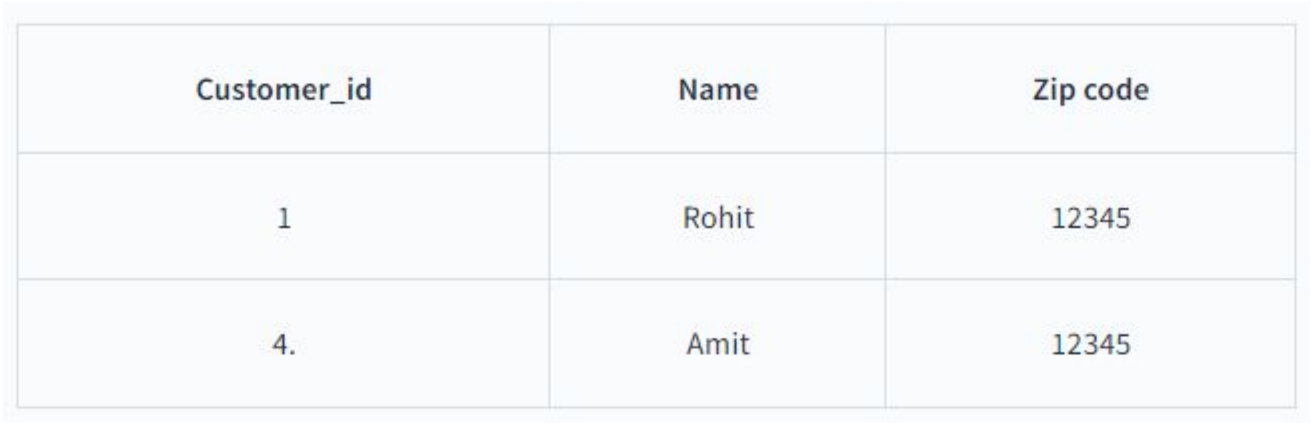
Write a TRC query to get all the data of customers whose zip code is 12345.

## Example 1

**TRC Query: {t \| t ∈ Customer ⋀ t.Zipcode = 12345} or TRC Query: {t \| Customer(t) ⋀ t[Zipcode] = 12345 }**

**Workflow of query** - The tuple variable "t" will go through every tuple of the Customer table. Each row will check whether the Cust_Zipcode is 12345 or not and only return those rows that satisfies the Predicate expression condition.

Result:



| Customer_id | Name | Zip code |
|---|---|---|
| 1 | Rohit | 12345 |
| 4. | Amit | 12345 |

## Example 1

Student(rollno, name, deptno, gender)

Find rollno and name of student in deptno 2.

**{t|t.rollno, t.name | student(t) ∧ t.deptno=2}**

Find rollno and name of male student in deptno 2.

**{t.rollno, t.name | student(t) ∧ t.deptno=2 ∧ t.gender=male}**

## Example 2

1. Find the names of all instructors whose department is in the watson building.

$\{ t \mid \exists s \in \text{instructor } (t[\text{name}]=s[\text{name}]) \wedge \exists u \in \text{department } (u[\text{deptname}]=s[\text{deptname}]) \wedge (u[\text{building}]=\text{“watson”}) \}$

2. Find the set of all courses taught in the fall 2009 semester or in the spring 2010 semester or both.

$\{ t \mid \exists s \in \text{section } (t[\text{courseid}]=s[\text{courseid}]) \wedge s[\text{semester}]=\text{“fall”} \wedge s[\text{year}]=2009 \vee \exists u \in \text{section } (t[\text{courseid}]=u[\text{courseid}]) \wedge u[\text{semester}]=\text{“spring”} \wedge u[\text{year}]=2010) \}$

## Domain Relational Calculus (DRC)

- The second form of relation is known as Domain relational calculus.
- In domain relational calculus, filtering variable uses the domain of attributes.
- Domain relational calculus uses the same operators as tuple calculus.
- It uses logical connectives $\wedge$ (and), $\vee$ (or) and $\neg$ (not).
- It uses Existential ($\exists$) and Universal Quantifiers ($\forall$) to bind the variable.

# Domain Relational Calculus (DRC)

**Notation:**

{ a1, a2, a3, ..., an | P (a1, a2, a3, ... ,an)}

Where, **a1, a2** are attributes

**P** stands for formula built by inner attributes or P(x1,x2,x3...) is predicate expression or condition.

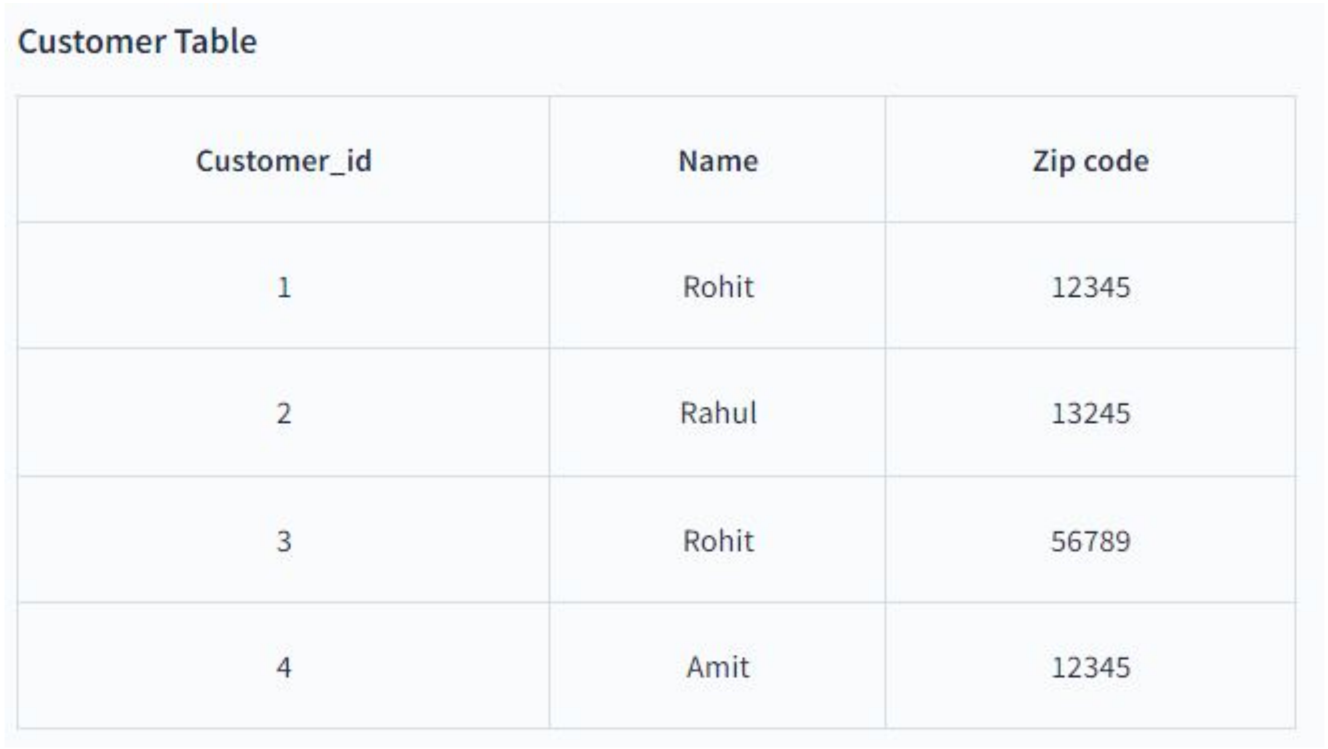
**For example:**

{< article, page, subject > | $\in$ Master $\wedge$ subject = 'database'}

**Output:** This query will yield the article, page, and subject from the relational Master, where the subject is a database.

## Unit 2
## Relational Model

## TRC and DRC



Customer Table

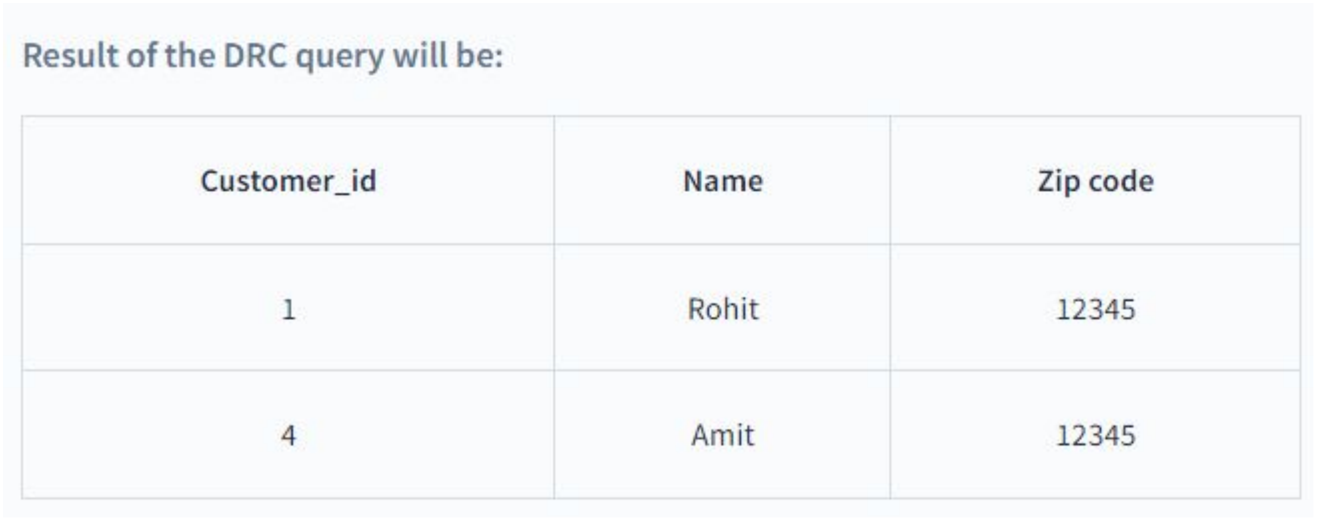
| Customer_id | Name | Zip code |
|---|---|---|
| 1 | Rohit | 12345 |
| 2 | Rahul | 13245 |
| 3 | Rohit | 56789 |
| 4 | Amit | 12345 |

Write a DRC query to get the data of all customers with Zip code 12345.

Unit 2
Relational Model

# TRC and DRC

- **DRC query: {<x1,x2,x3> \| <x1,x2> ∈ Customer ⋀ x3 = 12345 }**
- **Workflow of Query**: In the above query x1,x2,x3 (ordered) refers to the attribute or column which we need in the result, and the predicate condition is that the first two domain variables x1 and x2 should be present while matching the condition for each row and the third domain variable x3 should be equal to 12345.
- Result:



Result of the DRC query will be:

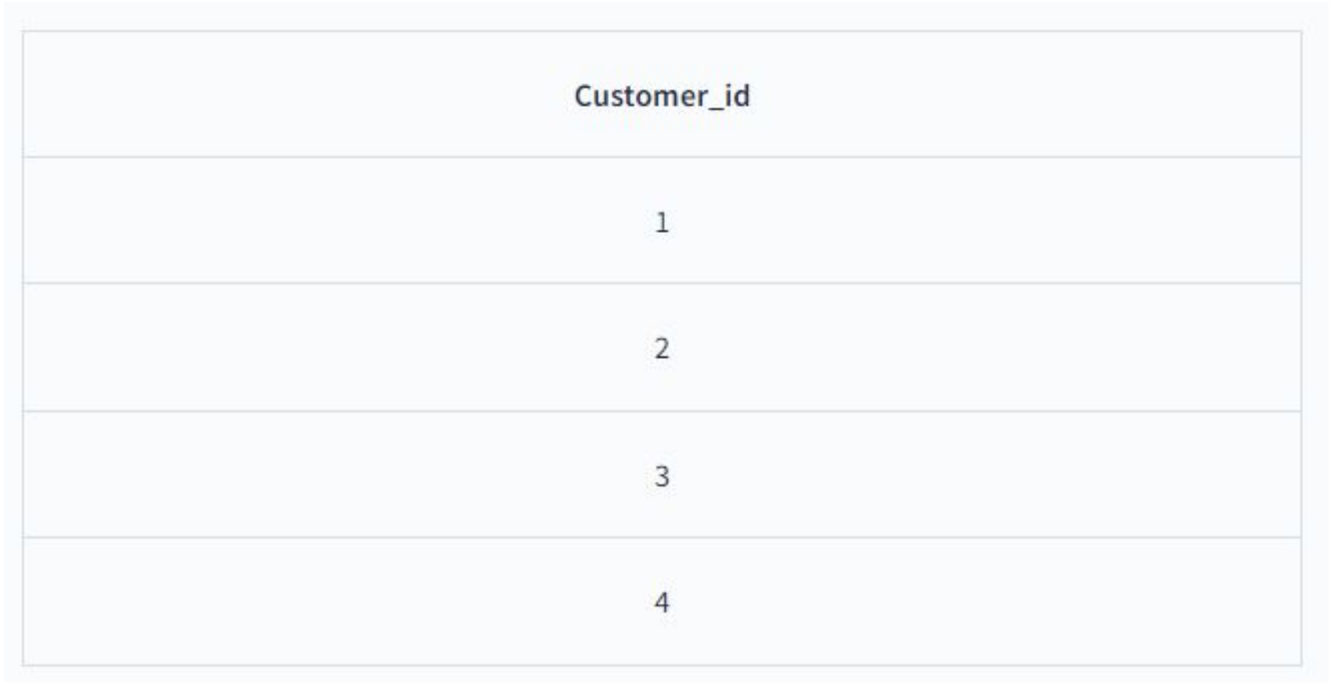
| Customer_id | Name | Zip code |
|---|---|---|
| 1 | Rohit | 12345 |
| 4 | Amit | 12345 |

Unit 2
Relational Model

# TRC and DRC

-  Write a DRC query to get the customer id of all the customer.

DRC Query: { <x1> \| ∃ x2,x3(<x1,x2,x3> ∈ Customer ) }



| Customer_id |
|---|
| 1 |
| 2 |
| 3 |
| 4 |

## TRC and DRC

DATABASE MANAGEMENT SYSTEM

DBMS

Unit 2
Relational Model

## TRC and DRC

For complete TRC and DRC information use below link.

https://www.youtube.com/watch?v=ekF4qQBsk18&index=1&list=PLx5CT0AzDJClCw2DlI7FYUcBn5MfvpZKY