Pune Vidyarthi Griha's
College of Engineering, Nashik
206, Behind Reliance Petrol Pump, Dindoriroad, Mhasrul, Nashik-422004
Phone: 0253-6480000 / 36 /44 Toll Free Number: 18002665330
Website: https://pvgcoenashik.org Email: admission@pvgcoenashik.org
Approved by AICTE, New Delhi, DTE, Mumbai and
Affiliated to Savitribai Phule Pune University, Pune.
DTE Code: EN5330

" Relational Model "

By,

Prof. Anand N. Gharu

Asst. Professor

Computer Department,

PVG COE, Nashik.

Introduction

- Relation Database 1 : https://youtu.be/2KCObY8ixgw
- Relational Database : https://youtu.be/NqdZnYZ7Gvw

RDBMS

Introduction

Relational Model :

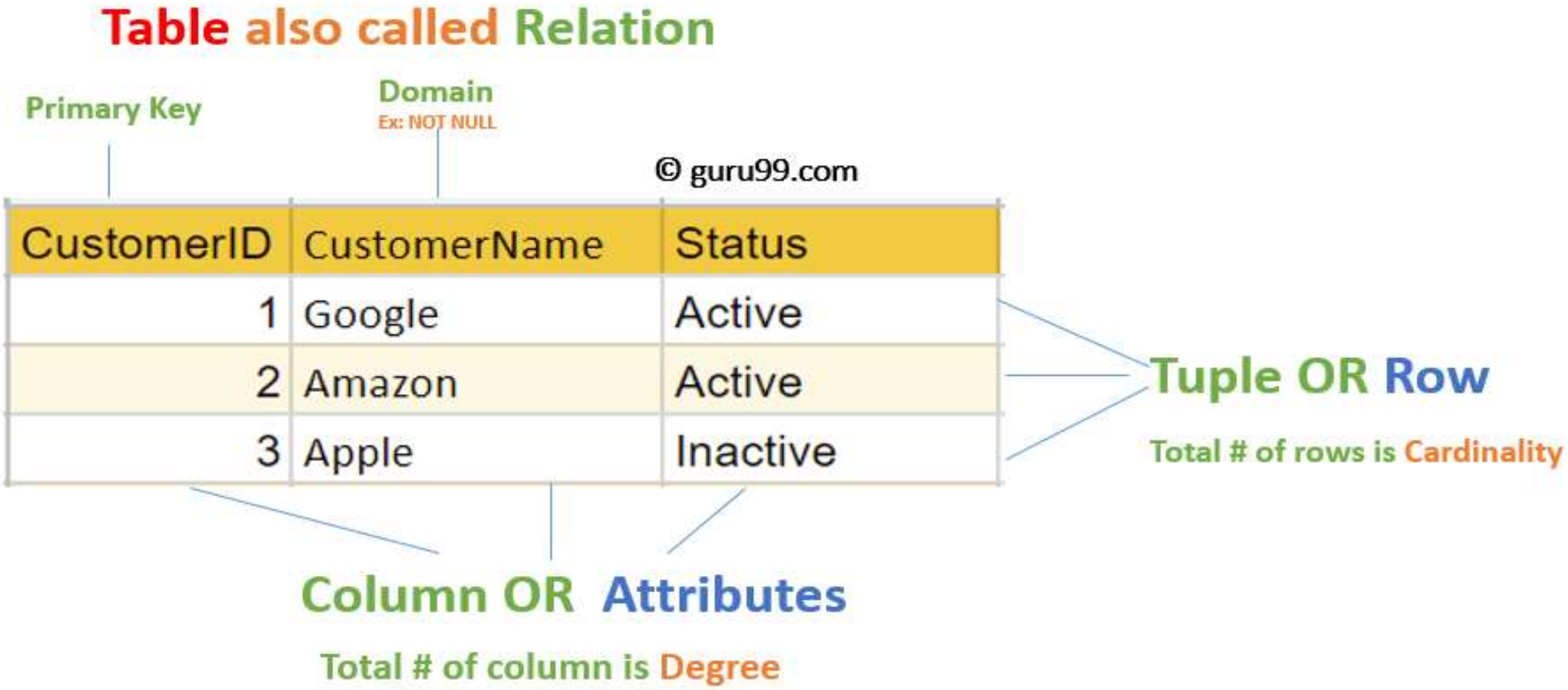
"Relational model can represent as a table with columns and rows. Each row is known as a tuple. Each table of the column has a name or attribute."

- Relational Model (RM) represents the database as a collection of relations.
- A relation is nothing but a table of values.
- Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

Some popular Relational Database management systems are:

1. DB2 and Informix Dynamic Server - IBM
2. Oracle and RDB – Oracle
3. SQL Server and Access - Microsoft

Relational Data Model Example



Advantages of using Relational Model

1. Relational Data Model(RDM) is easier to use
2. RDM is useful for small database.
3. RDM improves the performance of system because it is concerned with data.
4. Powerful DBMS
5. Relatively easy to understand
6. Easy to obtain information quickly
7. Reduced duplication
8. Support sharing of data

Advantages of using Relational Model

1. **Simplicity**:

   A Relational data model in DBMS is simpler than the hierarchical & network model.

1. **Structural Independence**: The relational database is only concerned with data and not with a structure. This can improve the performance of the model.
2. **Easy to use**: The Relational model in DBMS is easy as tables consisting of rows and columns are quite natural and simple to understand
3. **Query capability**: It makes possible for a high-level query language like SQL to avoid complex database navigation.
4. **Data independence**: The Structure of Relational database can be changed without having to change any application.
5. **Scalable**: Regarding a number of records, or rows, and the number of fields, a database should be enlarged to enhance its usability.

Disadvantages of using Relational Model

1. Hardware and system software overhead
2. Limited Data calculation
3. Physical storage consumption
4. Difficult in maintainance
5. Expertise required
6. Few relational databases have limits on field lengths which can't be exceeded.
7. Relational databases can sometimes become complex as the amount of data grows

Relational Data Model Concepts

Attribute: Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME,etc.

Tables – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.

Tuple – It is nothing but a single row of a table, which contains a single record.

Relation Schema: A relation schema represents the name of the relation with its attributes.

Data Dictionary in Database

Degree: The total number of attributes which in the relation is called the degree of the relation.

Cardinality: Total number of rows present in the Table.

Column: The column represents the set of values for a specific attribute.

Relation instance – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.

Relation key - Every row has one, two or multiple attributes, which is called relation key.

Attribute domain – Every attribute has some pre-defined value and scope which is known as attribute domain

Keys in RDM

“KEYS in DBMS is an attribute or set of attributes which helps you to identify a row(tuple) in a relation(table).”

Need of keys in DBMS :

- They allow you to find the relation between two tables.
- Keys help you uniquely identify a row in a table by a combination of one or more columns in that table.
- Key is also helpful for finding unique record or row from the table.
- Database key is also helpful for finding unique record or row from the table.

Types of keys in DBMS

1. **Super Key -** A super key is a group of single or multiple keys which identifies rows in a table.
2. **Primary Key -** is a column or group of columns in a table that uniquely identify every row in that table.
3. **Candidate Key -** is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.
4. **Alternate Key -** is a column or group of columns in a table that uniquely identify every row in that table.

Types of keys in DBMS

4. Foreign Key - is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.

5. Compound Key - has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database.

6. Composite Key - An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key.

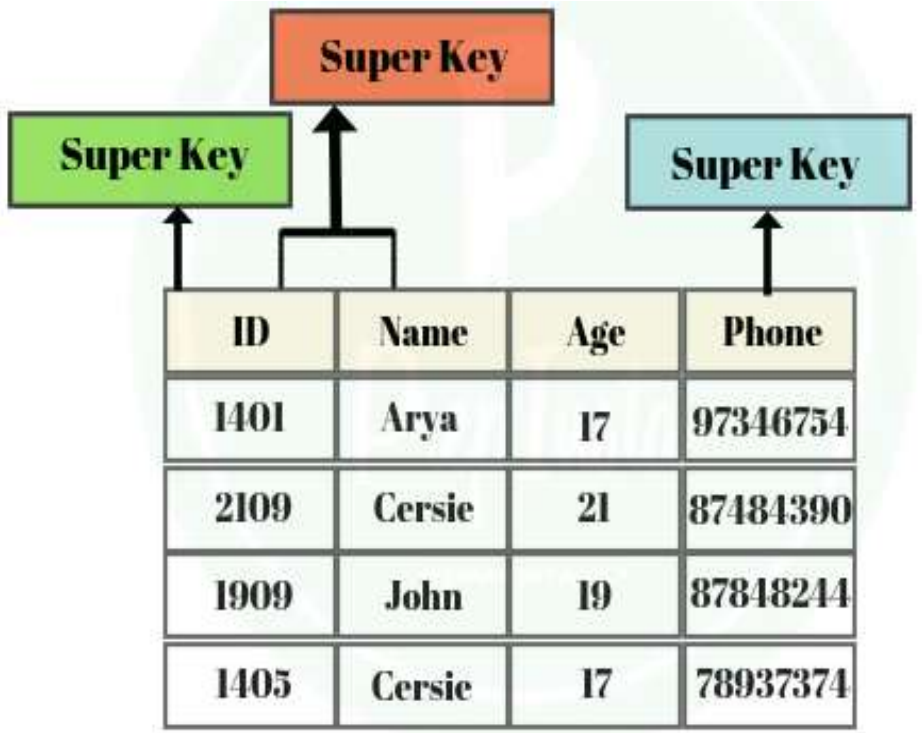
7. Surrogate Key - An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key.

Super Key

Super Key: A super key is a set of one or more attributes (columns), which can uniquely identify a row in a table.

For Example, STUD_NO, (STUD_NO, STUD_NAME) etc.

- Adding zero or more attributes to candidate key generates super key.
- A candidate key is a super key but vice versa is not true.



ID	Name	Age	Phone
1401	Arya	17	97346754
2109	Cersie	21	87484390
1909	John	19	87848244
1405	Cersie	17	78937374

Primary Key

PRIMARY KEY is a column or group of columns in a table that uniquely identify every row in that table. The Primary Key can't be a duplicate meaning the same value can't appear more than once in the table. A table cannot have more than one primary key.

Rules for defining Primary key:

- Two rows can't have the same primary key value
- It must for every row to have a primary key value.
- The primary key field cannot be null.
- The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.

Primary Key

Example:

In the following example, **StudID is a Primary 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

Alternate Key

ALTERNATE KEYS is a column or group of columns in a table that uniquely identify every row in that table. A table can have multiple choices for a primary key but only one can be set as the primary key. All the keys which are not primary key are called an Alternate 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.

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

Candidate Key

CANDIDATE KEY is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes. The Primary key should be selected from the candidate keys. Every table must have at least a single candidate key. A table can have multiple candidate keys but only a single primary key.

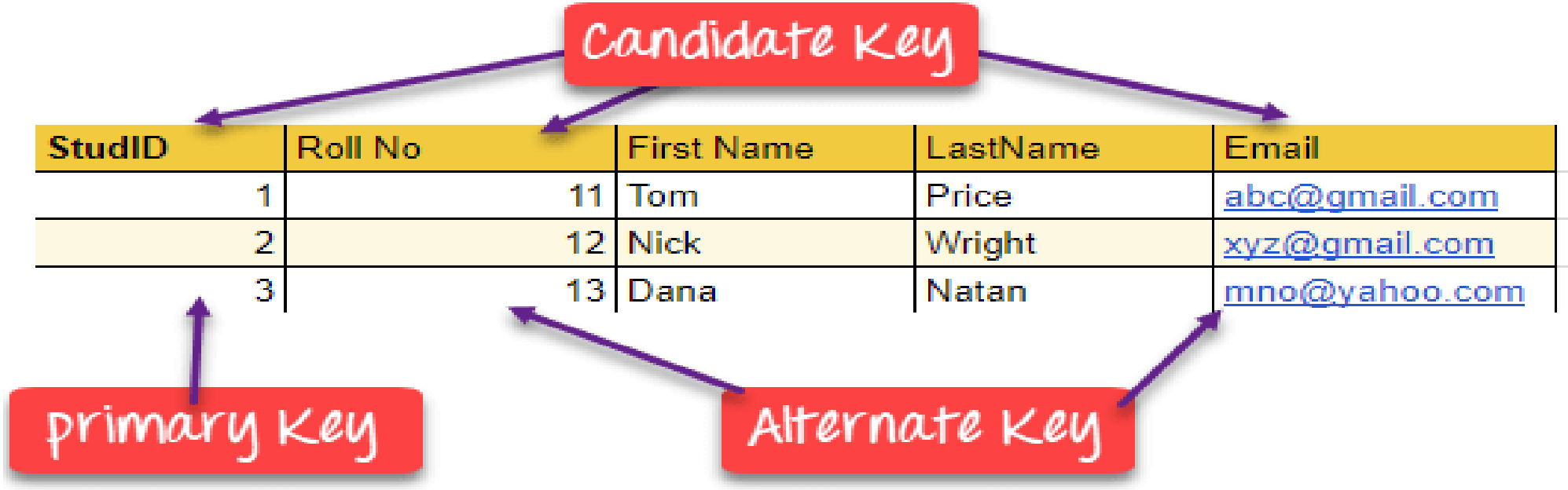
Properties of Candidate key:

- 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

Candidate Key

Example:

In the given table Stud ID, Roll No, and email are candidate keys which help us to uniquely identify the student record in the table.



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

Foreign key

FOREIGN KEY is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity. It acts as a cross-reference between two tables as it references the primary key of another table.

Example:

DeptCode	DeptName
001	Science
002	English
005	Computer

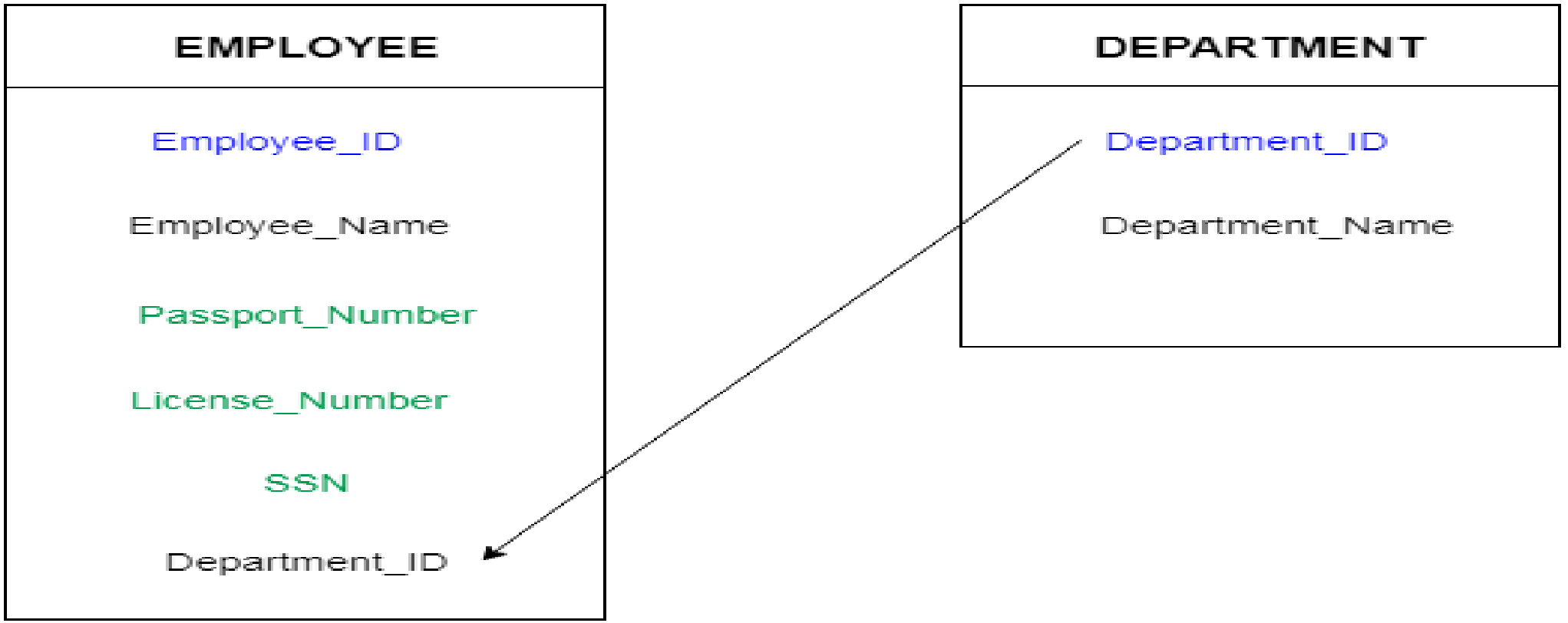
Teacher ID	Fname	Lname
B002	David	Warner
B017	Sara	Joseph
B009	Mike	Brunton

Foreign key

We add the primary key of the DEPARTMENT table, Department_Id as a new attribute in the EMPLOYEE table.

Now in the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.

This concept is also known as Referential Integrity.



Compund key

COMPOUND KEY has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database. However, when combined with the other column or columns the combination of composite keys become unique. The purpose of the compound key in database is to uniquely identify each record in the

Example:

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

Composite key

COMPOSITE KEY is a combination of two or more columns that uniquely identify rows in a table. The combination of columns guarantees uniqueness, though individually uniqueness is not guaranteed. Hence, they are combined to uniquely identify records in a table.

- The difference between compound and the composite key is that any part of the compound key can be a foreign key, but the composite key may or maybe not a part of the foreign key

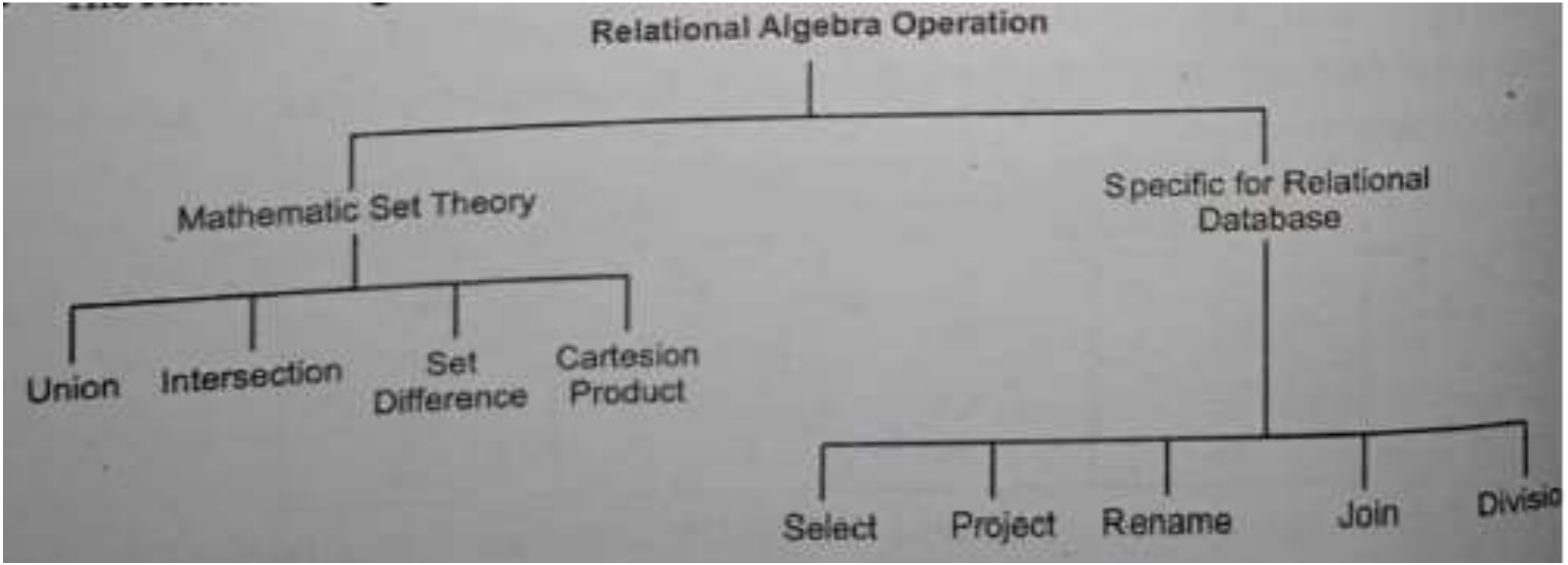
OrderNo	PorductID	Product Name	Quantity
B005	JAP102459	Mouse	5
B005	DKT321573	USB	10
B005	OMG446789	LCD Monitor	20
B004	DKT321573	USB	15
B002	OMG446789	Laser Printer	3

Difference between primary & foreign key

Primary Key	Foreign Key
1. Only one primary key in table	1. More than one foreign key in table
2. Primary key uniquely identify a record in the table.	2. It is a field in the table that is the primary key of another table.
3. Primary Key never accept null values.	3. A foreign key may accept multiple null values.
4. Primary key does not allow duplicate value	4. Foreign key allow duplicate value
5. Primary key is a clustered index	5. A foreign key cannot automatically create an index, clustered or non-clustered.
6. Its value cannot be deleted from the parent table.	6. Its value can be deleted from the child table.

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"



Relational Algebra

- **Unary Relational Operations**

SELECT (symbol: σ)

PROJECT (symbol: π)

RENAME (symbol: ρ)

- **Binary Relational Operations**

JOIN

DIVISION

- **Relational Algebra Operations From Set Theory**

UNION (υ)

INTERSECTION ( ),

DIFFERENCE (-)

CARTESIAN PRODUCT (x)

Relational Algebra Operations

1. Select Operator (σ) :

Select Operator is denoted by sigma (σ) and it is used to find the tuples (or rows) in a relation (or table) which satisfy the given condition.

If you understand little bit of SQL then you can think of it as a where clause in SQL, which is used for the same purpose.

Syntax of Select Operator (σ)

σ Condition/Predicate(Relation/Table name)

Relational Algebra Operations

1. Select Operator (σ) :

Select Operator (σ) Example

Table: CUSTOMER

Customer_Id	Customer_Name	Customer_City
C10100	Steve	Agra
C10111	Raghu	Agra
C10115	Chaitanya	Noida
C10117	Ajeet	Delhi
C10118	Carl	Delhi

Select Operator (σ) :

Query:

σ Customer_City="Agra" (CUSTOMER)

Output:

Customer_Id	Customer_Name	Customer_City
C10100	Steve	Agra
C10111	Raghu	Agra

Relational Algebra Operations

2. Project Operator (∏)

Project operator is denoted by ∏ symbol and it is used to select desired columns (or attributes) from a table (or relation).

Project operator in relational algebra is similar to the Select statement in SQL.

Syntax of Project Operator (∏)

∏ column_name1, column_name2, ..., column_nameN(table_name)

Relational Algebra Operations

Project Operator (∏) Example

In this example, we have a table CUSTOMER with three columns, we want to fetch only two columns of the table, which we can do with the help of Project Operator ∏.

```
Table: CUSTOMER

Customer_Id      Customer_Name       Customer_City
-----------      -------------       -------------
C10100            Steve               Agra
C10111            Raghu               Agra
C10115            Chaitanya           Noida
C10117            Ajeet               Delhi
C10118            Carl                Delhi
```

Query:

```
∏ Customer_Name, Customer_City (CUSTOMER)
```

Output:

```
Customer_Name       Customer_City
-------------       -------------
Steve               Agra
Raghu               Agra
Chaitanya           Noida
Ajeet               Delhi
Carl                Delhi
```

Relational Algebra Operations

3. Union Operator (∪)

Union operator is denoted by ∪ symbol and it is used to select all the rows (tuples) from two tables (relations).

For a union operation to be valid, the following conditions must hold -

1. R and S must be the same number of attributes.
2. Attribute domains need to be compatible.
3. Duplicate tuples should be automatically removed.

Syntax of Union Operator (∪)

table_name1 ∪ table_name2

Relational Algebra Operations

3. Union Operator (∪)

Query:

```
∏ Student_Name (COURSE) ∪ ∏ Student_Name (STUDENT)
```

Output:

```
Student_Name
-----------
Aditya
Carl
Paul
Lucy
Rick
Steve
```

Union Operator (∪) Example

Table 1: COURSE

Course_Id	Student_Name	Student_Id
C101	Aditya	S901
C104	Aditya	S901
C106	Steve	S911
C109	Paul	S921
C115	Lucy	S931

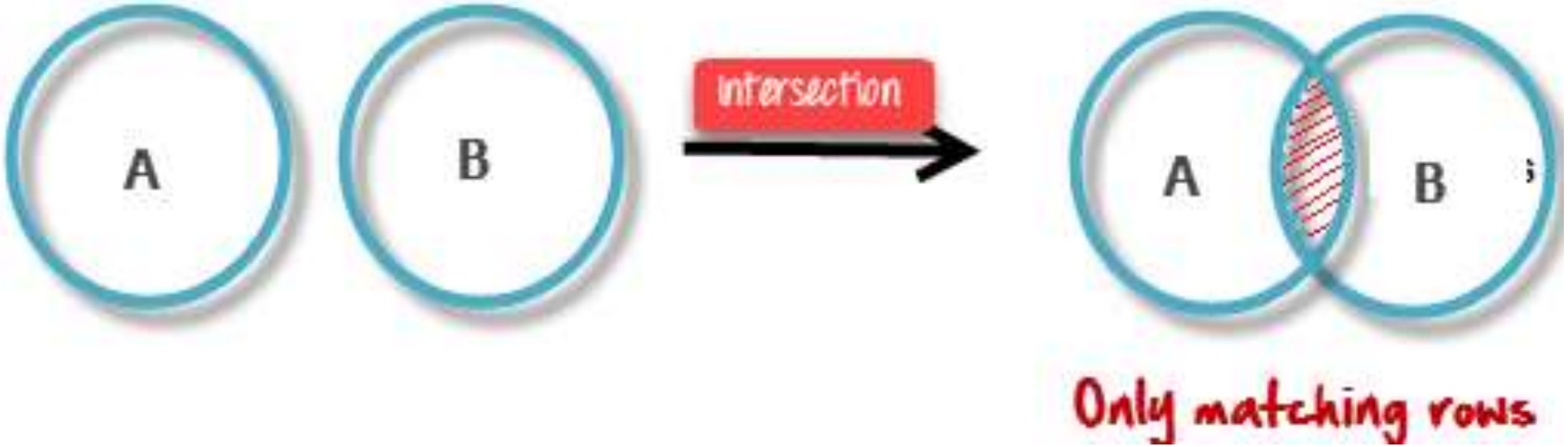
Table 2: STUDENT

Student_Id	Student_Name	Student_Age
S901	Aditya	19
S911	Steve	18
S921	Paul	19
S931	Lucy	17
S941	Carl	16
S951	Rick	18

Relational Algebra Operations

Intersection Operator (∩)

Intersection operator is denoted by ∩ symbol and it is used to select common rows (tuples) from two tables (relations).



Note: Only those rows that are present in both the tables will appear in the result set.

Syntax of Intersection Operator (∩)

table_name1 ∩ table_name2

Relational Algebra Operations

Intersection Operator (∩)

Query:

```
∏ Student_Name (COURSE) ∩ ∏ Student_Name (STUDENT)
```

Output:

```
Student_Name
------------
Aditya
Steve
Paul
Lucy
```

Table 1: COURSE

Course_Id	Student_Name	Student_Id
C101	Aditya	S901
C104	Aditya	S901
C106	Steve	S911
C109	Paul	S921
C115	Lucy	S931

Table 2: STUDENT

Student_Id	Student_Name	Student_Age
S901	Aditya	19
S911	Steve	18
S921	Paul	19
S931	Lucy	17
S941	Carl	16
S951	Rick	18

Relational Algebra Operations

Set Difference (-)

Set Difference is denoted by – symbol. Lets say we have two relations R1 and R2 and we want to select all those tuples(rows) that are present in Relation R1 but **not** present in Relation R2, this can be done using Set difference R1 – R2.

Syntax of Set Difference (-)

```
table_name1 - table_name2
```

Set Difference (-) Example

Lets take the same tables COURSE and STUDENT that we have seen above.

Query:

Lets write a query to select those student names that are present in STUDENT table but not present in COURSE table.

```
π Student_Name (STUDENT) - π Student_Name (COURSE)
```

Output:

```
Student_Name
------------
Carl
Rick
```

Relational Algebra Operations

Cartesian product (X)

Cartesian Product is denoted by X symbol. Lets say we have two relations R1 and R2 then the cartesian product of these two relations (R1 X R2) would combine each tuple of first relation R1 with the each tuple of second relation R2. I know it sounds confusing but once we take an example of this, you will be able to understand this.

Syntax of Cartesian product (X)

```
R1 X R2
```

Cartesian product (X) Example

Table 1: R

```
Col_A     Col_B
-----     ------
AA          100
BB          200
CC          300
```

Table 2: S

```
Col_X      Col_Y
-----      -----
XX           99
YY           11
ZZ           101
```

Relational Algebra Operations

Query:

Lets find the cartesian product of table R and S.

```
R X S
```

Output:

```
Col_A    Col_B     Col_X     Col_Y
-----    ------    ------    ------
AA        100       XX        99
AA        100       YY        11
AA        100       ZZ        101
BB        200       XX        99
BB        200       YY        11
BB        200       ZZ        101
CC        300       XX        99
CC        300       YY        11
CC        300       ZZ        101
```

Note: The number of rows in the output will always be the cross product of number of rows in each table. In our example table 1 has 3 rows and table 2 has 3 rows so the output has 3×3 = 9 rows.

Relational Algebra Operations

Rename (ρ)

Rename (ρ) operation can be used to rename a relation or an attribute of a relation.

Rename (ρ) Syntax:

ρ(new_relation_name, old_relation_name)

'rename' operation is denoted with small Greek letter **rho** ρ.

Rename (ρ) Example

Lets say we have a table customer, we are fetching customer names and we are renaming the resulted relation to CUST_NAMES.

Table: CUSTOMER

Customer_Id	Customer_Name	Customer_City
C10100	Steve	Agra
C10111	Raghu	Agra
C10115	Chaitanya	Noida
C10117	Ajeet	Delhi
C10118	Carl	Delhi

Relational Algebra Operations

Rename Operator (∩)

Query:

```
ρ(CUST_NAMES, Π(Customer_Name)(CUSTOMER))
```

Output:

```
CUST_NAMES
----------
Steve
Raghu
Chaitanya
Ajeet
Carl
```

Relational Algebra Operations

nd certain logical operates shown in Fig. 3.15:

Operator	Meaning	Example
=	equal to	name = "Atul"
< >	not equal to	city < > "Pune"
>	greater than	age > 20
>=	greater than equal to	salary >= 20000
<	less than	profit < 1000
<=	less than equal to	marks <= 35

Fig. 3.15: Comparison Operators

Operator	Meaning	Example
AND	Logical AND	name = "Atul" AND city = "Pune"
OR	Logical OR	age = 18 OR age <= 42

Relational Algebra Operations

Example 3.1: Let's consider a database where Branch, Account, Customer and Loan are the relations available. There are many accounts in a branch. Customers and loans are related with Many-to-Many relationship. Similarly, Customer and Account are related with Many-to-Many. A branch can have many loans. The E-R diagram is as shown below in Fig. 3.36.

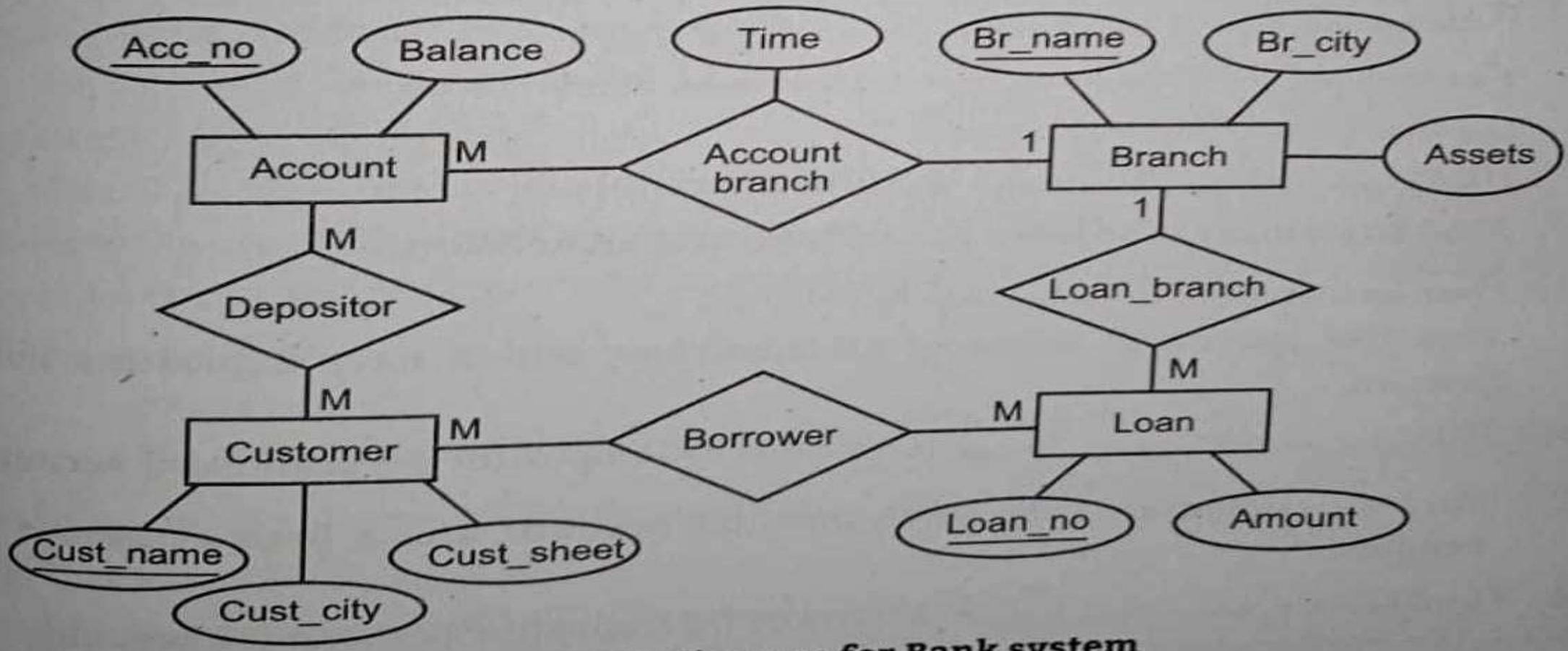


Fig. 3.36: E-R diagram for Bank system

for the following questions.

Relational Algebra Operations

Now, let's write the relational algebra expressions for the following

1. **Select a record from relation Loan where branch is Jhons Street.**

Sol.: $\sigma_{br\_name = \text{"Jhons Street"}}$ (Loan)

2. **Find name of all branches in the loan relation.**

Sol.: Π_{br_name} (Loan)

3. **Find the branch in the loan relation where branch is Jhons Street and Loan amount is greater than 2,00,000**

Sol.: Π_{br_name} ($\sigma_{br\_name = \text{"Jhons Street"} \cap amt > 200000}$ (Loan))

4. **Find names of all customers who have either a loan or an account.**

Sol.: Π_{cust_name} (Borrower) $\cup$ Π_{cust_name} (Depositor)

5. **Find all customers of the bank who have an account but not a loan.**

Sol.: Π_{cust_name} (Depositor) – Π_{cust_name} (Borrower)

Relational Algebra Operations

Sol.: Π_{cust_name} (Depositor) ∩ Π_{cust_name} (

6. Find the name of all customers who have loan at Adams Street branch.

Sol.: Here, we need the information of both the relations i.e. Loan and Borrower. Because loan_no is the common attribute available so the common loan_no of Adams Street branch should get selected.

$$\Pi_{cust\_name}(\sigma_{borrower.loan\_no = loan.loan\_no}(\sigma_{br\_name = \text{"Adams Street"}}(Borrower \times Loan)))$$

7. Find the names of all customers who live on the same street and in the same city as Mac.

Sol.: $\Pi_{cust\_street, cust\_city}(\sigma_{cust\_name = \text{"Mac"}}(Customer))$

But to find other customer with this street and city, we must reference the customer relation second time.

$$\Pi_{customer.cust\_name}$$

$$(\sigma_{customer.cust\_street = dummy.cust\_street \cap customer.cust\_city = dummy.cust\_city}$$

$$(customer \times \rho_{dummy\ (street, city)}$$

$$(\Pi_{cust\_street, cust\_city}(\sigma_{cust\_name = \text{"Mac"}}(Customer)))))$$

Relational Algebra Operations

8. **Find customers who have both a loan and an account.**

Sol.: $\Pi_{cust\_name}(borrower) \cap \Pi_{cust\_name}(Depositor)$.

9. **Find the asset and name of all branches, which have depositors living in London.**

Sol.: $\Pi_{br\_name, assets}(\sigma_{cust\_city = "London"}(Customer \bowtie Depositor \bowtie Branch \bowtie Account)$

10. **Find all customers who have both an account and a loan at Bezant Street branch.**

Sol.: $\Pi_{cust\_name}(\sigma_{br\_name = "Bezant Street"}(Borrower \bowtie Depositor \bowtie Loan \bowtie Account)$

Relational Algebra Operations

Example 3.2: Consider the following relational database:

DOCTOR (Doctor_no, Doctor_name, address, City)

Hospital (Hosp_no. hosp_name, street , H_city)

Doct_Hosp (Doctor_nao, hosp_no, Date)

Construct following queries into relational algebra

1: Find out all the doctors who have visited to hospitals in the same city in which they live.

Sol.: $\Pi_{doctor\_name}(\sigma_{doctor.city = hospital.H\_city}$

(DOCTOR ⋈ Hostipal ⋈ Doct_hosp))

2.: Find out to which hospital Dr. Will Smith has visited.

Sol.: $\Pi_{hosp\_name}(\sigma_{doctor\_name = \text{"Dr. Will Smith"}}$

(Doctor ⋈ Hospital ⋈ Doct_hosp))

Relational Algebra Operations

Example 3.4: An orchestra database consists of the following database.

Conducts (conductor, composition)

Requires (composition, instrument)

Plays (player, instrument)

1.: List the composition conducted by Bran Adam who do not use violin.

Sol.: $\Pi_{composition}(\sigma_{conductor = \text{"Bran Adam"} \wedge instrument \Leftrightarrow \text{"Violin"}}$

$(Conducts \bowtie Requires))$

2.: List the players which are likely to play for the composition 'Last Symphony'.

Sol.: $\Pi_{player}(\sigma_{composition = \text{"Last Symphony"}}(Requires \bowtie Plays))$.

Relational Algebra Operations

Example 3.5: Consider the following relational databases.

Student (S_no, S_name, B_data, Class)

Course (C_no, C_name)

Result (S_no, C_no, grade)

Construct the following queries in Relational Algebra.

Construct the following queries in Relational Algebra.

(a) Find out student names who have secured grade 'A' in course 'DBMS'.

Sol. $\pi_{student.s\_name}$

$(\sigma_{result.grade = "A"}$

$\sigma_{course.c\_name = "DBMS"}$ (Student ⋈ Course ⋈ Result))

(b) Find out the grades of students 'Roger' along with course name.

Sol. $\pi_{result, grade, course.c\_name}$

$(\sigma_{student.s\_name = "Roger"}$ (Student ⋈ Course ⋈ Result))

Relational Algebra Operations

Example 3.6: The student database consists of the following database.

Student (Stud_no, S_name)

Teach (Prof, C_code, section)

Guides (Prof, Stud_no)

Enroll (Stude_no, c_code, section)

Construct the following queries into Relational Algebra.

1. List all student taking course with prof. "Adam".

Sol. $\Pi_{s\_name, stud\_no}(\sigma_{prof = "Adam"}(Enroll \bowtie Teach \bowtie Student))$

2. List all student who are guided by the project guide of the student named Alex.

Sol. $Guides \bowtie \Pi_{prof}(\Pi_{stud\_no}(\sigma_{s\_name = "Alex"}(Guides \bowtie Student)))$

Relational Algebra Operations

(*b*) Consider relational database : [8]

Customer (cust-no, cust-name, address, city)

Loan (loan-no, loan-amt, loan-date, cust-no)

Customer and loan are related with one to many relationships.

Write relational algebraic expression of the following :

(*i*) Display customer with loan amount greater than 1,00,000.

(*ii*) List names of customer who do not have loan at the bank.

(*iii*) List loan details of customer name as "Mr. Dhumale".

(*iv*) List of the customer who have taken loan from the bank with amount more than 50,000 and city as 'Pune'.

Relational Algebra Operations

(*b*) Consider the database and write relational algebraic expression

Patient Master (PatientNo. PatientName, Sex, Address City, Allergy, Chief Complaints)

(*i*) Display all patients whose Allergy is "Nimesulide"

(*ii*) Display all male patients from city Calcutta.

(*iii*) Update all patients whose sex is "M" with "Male".

(*iv*) List all patients whose chief complaint is "fever".

Relational Algebra Operations

(*b*) Consider relational database : [8]

Supplier (Supno, sname, supaddress)

Item (Itemno, Iname, stock)

Supp-Item (Supno, Itemno, rate)

Write relational algebraic expression for the following :

(*a*) List all suppliers from 'Varanasi' city who supplies PISTON.

(*b*) Display all suppliers supply PISTON RINGS

(*c*) Change supplier names to upper case

(*d*) List all suppliers supplying DOOR Lock from 'Jaipur' city.

THANK YOU!!!

My Blog : https://anandgharu.wordpress.com/

Email : gharu.anand@gmail.com