Chapter 3
Relational Data Model

Relational Model

Relational Model was proposed by E.F. Codd to model data in the form of relations or tables.

After designing the conceptual model of Database using ER diagram, we need to convert the conceptual model in the relational model. The relational model of data is based on the concept of relation (table). Relational Model represents how data is stored in Relational Databases. A relational database stores data in the form of relations (tables).

Relational Database Management System (RDBMS)

A relational database is a database structured in relational model. A RDBMS is a collection of programs that can be used to create, maintain, modify and manipulate the relational database. Some of the example of RDBMS are Oracle, MS-SQL Server , MS-Access etc.

Properties of RDBMS

1. Provides data to be stored in tables
2. Persists data in the form of rows and columns
3. Provides facility primary key, to uniquely identify the rows
4. Creates indexes for quicker data retrieval
5. Provides a virtual table creation in which sensitive data can be stored and simplified query can be applied.(views)
6. Sharing a common column in two or more tables(primary key and foreign key)
7. Provides multi user accessibility that can be controlled by individual users.

Advantages of Relational Model

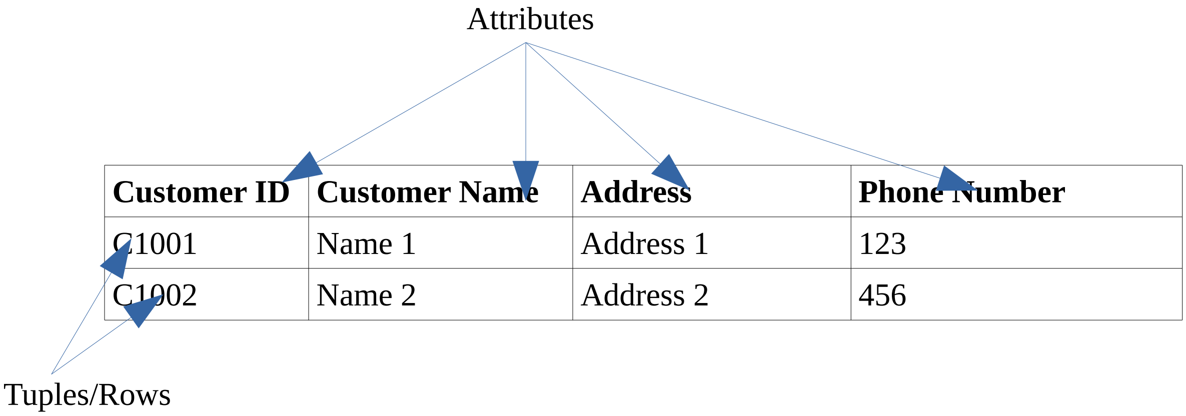
1. It is very easy to use data are stored in the form of rows and column which is easy to perceive.
2. It is a very flexible approach.
3. It is secured. We can keep the sensitive attributes in separate relation with its own authorization control.
4. Data independence is another feature of this approach.
5. The query based on relational algebra for accessing the data is easy.

Disadvantages of Relational Model

If the numbers of tables between which relationship is to be established are large and voluminous, the performance in responding to queries slows down.

Structure of Relational Model

A relational database consists of collections of tables, each of which is assigned a unique name. A row in a table represents a relationship among a set of values. A table is a collection of such relationships.



Customer ID	Customer Name	Address	Phone Number
C1001	Name 1	Address 1	123
C1002	Name 2	Address 2	456

The above table consists of below listed components according to relational model.

1. Domain:

   A set of permitted values for an attributes.
2. Tuple:

   A record/row in a table is called tuple. Every relation is made up of many tuples.
3. Relation:

   The term relation in this model refers to the two-dimensional table of data.
4. Relation Schema:

   A relation schema consists of a list of attributes and their corresponding domains.
5. Relation instance:

   The relation instance is the collection of data at any instance of time within a relation.

Q. Why a database is called as relational database model?

A database model represents the relationship between one or more databases. The relationship is known as the relational database model. It is an extension of the normal databases without relations.

It provides flexibility and allows one database to be in relation with another database. It can access the data from many databases at one time over the network.

Difference between DBMS and RDBMS

DBMS	RDBMS
DBMS applications store data as file.	RDBMS applications store data in a tabular form.
In DBMS, data is generally stored in either a hierarchical form.	In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.
Normalization is not present in DBMS.	Normalization is present in RDBMS.
DBMS uses file system to store data, so there will be no relation between the tables.	In RDBMS, data values are stored in the form of tables, so a relationship between these data values will be stored in the form of a table as well.
DBMS has to provide some uniform methods to access the stored information.	RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information.
DBMS does not support distributed database.	RDBMS supports distributed database.
DBMS is meant to be for small organization and deal with small data. it supports single user.	RDBMS is designed to handle large amount of data. It supports multiple users.
Examples of DBMS are file systems, xml etc.	Example of RDBMS are mysql, postgre, sql server, oracle etc.

Kinds of Relation (Table)

Basically there are three types of relations. These are

1. Base tables

   A table that physically exist in the database with unique name. It can be created, altered and manipulated using SQL statement.

2. Query Result table:

   A table that is displayed to the user for showing the resultant data obtained when a ‘question’ is asked to a table.

3. Views:

   A view is a virtual table which consists of only those columns from base table that the database user wants to see.

Relational Algebra

The relational algebra is a procedural language which consists of set of operations that take one or two relations as input and produce a new relation as a result. Relation algebra can be viewed as data manipulation language for relation model.

The fundamental operations in relation algebra are

1. Select
2. Project
3. Union
4. Set Difference
5. Rename
6. Cartesian product.

More three operations have been defined in term of these six, they are

1. Set Intersection
2. Joins
3. Division.

1. Selection Operator

Selection operator is unary operator. The selection operator is sigma: **σ**. The selection operator acts like filter on a relation by returning only a certain numbers of tuples. The resulting relation will have the same degree as the original relation. However the number of tuples is less than that of original relation.

$\boldsymbol{\sigma_C(R)}$ returns all those tuple in relation R that satisfy the condition C. A condition C can be made up of any combination of comparison or logical operator that operates on the attribute of R.

Comparison operators: **= , <, >, ≥, ≤, ≠**

Logical operators: **¬, ∧, ∨ : (**not, and , or) repspectively

Example:

Assume the following relation tbl_employee has the following attributes and tuples.

Name	Office	Dept	Rank
Smith	400	CS	Assistant
Jones	220	Eco	Adjunct
Greens	160	Eco	Assistant
Brown	420	CS	Associate
Smith	500	Fin	Associate

1. Select those employees in the CS department.

$$\boldsymbol{\sigma_{Dept='CS'}(tbl\_employee)}$$

Name	Office	Dept	Rank
Smith	400	CS	Assistant
Brown	420	CS	Associate

2. Select those employees with name Smith and who are assistant.

$$\boldsymbol{\sigma_{Name='Smith' \wedge Rank='Assistant'}(tbl\_employee)}$$

Name	Office	Dept	Rank
Smith	400	CS	Assistant

3. Select those employees who are either Assistant or in the Economic department.

$$\boldsymbol{\sigma_{Rank='Assistant' \vee Dept='Econ'}(tbl\_employee)}$$

Name	Office	Dept	Rank

Smith	400	CS	Assistant
Jones	220	Eco	Adjunct
Greens	160	Eco	Assistant

2. Projection Operator

Projection is also a unary operator. The projection operator is pi: $\prod$. Projection limits the attributes that will be returned from the original relation.

The general syntax is $\prod_{\text{attribute}}(R)$ where attribute is the list of attributes to be displayed from the original relation R. The resulting relation have the same number of tuples as the original relation however the degree of the resulting relation may be equal to or less than that of the original relation.

Example: project only name and department of the employees.

$$\prod_{\text{Name, Dept}} \text{(tbl\_employee)}$$

Name	Dept
Smith	CS
Jones	Eco
Greens	Eco
Brown	CS
Smith	Fin

Combining Selection and projection

The selection and projection operators can be combined to perform both operation i.e. display selected tuples with selected attributes.

Example:

1. Show the Name of all employees working in CS Dept.

$$\prod_{\text{Name(}}\sigma_{\text{Dept='CS'}}\text{(tbl\_employee)}$$

Name
Smith
Brown

2. Show the Name and Rank of those Employees who are not in CS department or Adjuncts.

$$\prod_{\text{Name, Rank}} (\sigma_{\text{Dept}\neq\text{'CS'} \wedge \text{Rank}\neq\text{'Adjunct'}}(\text{tbl\_employee}))$$

Or

$$\prod_{\text{Name, Rank}} (\sigma_{\neg\ (\text{Dept='CS'} \vee \text{Rank='Adjunct'})}(\text{tbl\_employee}))$$

Name	Rank
Greens	Eco
Smith	Fin

Note: $\neg(a \vee b) = \neg a \wedge \neg b$ ***and*** $\neg(a \wedge b) = \neg a \vee \neg b$ ***(De-Morgans law)***

3. Union

The union operation on two relations R and S denoted by R U S having same number of attributes results a new relation having records from both relation with duplicate records removed.

4. Set Difference

The difference operation on two relations R and S denoted by R-S results a relation containing those records from R but not from S.

5. Intersection

The intersection operation on two relations R and S denoted by R∩S results a relation with records that appears in both R and S.

Example: Consider the following two relation

R

Name	Dept
Ram	Mgmt
Jones	Eco

S

Name	Dept
Smith	CS
Jones	Eco
Greens	Eco
Brown	CS
Smith	Fin

1. R U S

Name	Dept
Ram	Mgmt
Jones	Eco
Smith	CS
Greens	Eco
Brown	CS
Smith	Fin

2. R – S

Name	Dept
Ram	Mgnt

3. R ∩ S

Name	Dept
Jones	Eco

6. Rename Operator (ϱ)

The rename operation on and existing relations results a relation under a new name.

$\varrho_A(B)$ is the relation with B with its name changed to A.

7. Cartesian Product

The Cartesian product also called as cross product or cross join is used to produce all combination of records from two relations i.e. it combines the record on one relation with all the records from another relation.

If R and S are two relation having n and m number of attributes and p and q number of records respectively , then the Cartesian product of these two relation denoted by R X S results a new relation having (n+m) number of attributes and (p*q) number of records.

R

Launch	Time
Momo	02:00:00 PM
Chowmin	01:00:00 PM

S

ID	Name
1	Ram
2	Hari
3	Sita
4	Gita

R X S

ID	Name	Lunch	Time
1	Ram	Momo	02:00:00 PM
1	Ram	Chaumin	01:00:00 PM
2	Hari	Momo	02:00:00 PM
2	Hari	Chaumin	01:00:00 PM
3	Sita	Momo	02:00:00 PM
3	Sita	Chaumin	01:00:00 PM
4	Gita	Momo	02:00:00 PM
4	Gita	Chaumin	01:00:00 PM

8. Division

The divide operator takes two relations and builds another relation consisting of values of an attribute of one relation that matches all the values in another relation.

R

Subject Name	Course
DBMS	Cmp
C++	Elx
C++	Cmp
OS	Cmp

S

Course
Cmp

R÷ S

Subject name
C++

9. Joins

Joins operation bring together two relations and combine their attributes and tables in a specific fashion.

Join is a combination of a Cartesian product followed by a selection process. In its simplest form, the join operation is just the cross product of two relations which produce large result size but using this operation, one record from relation R and one record from Relation S can be combined together to form the result if the combination satisfies the join condition. The join condition can be =, <, >, ≤, ≥, ≠

A. Equijoin

A join when the join condition is = then we call it Equi-join.

Example: Assume we have two relations EMP and DEPART as below.

EMP

Name	Office	Dept	Salary
Smith	400	CS	10000
Jones	220	Econ	15000
Green	100	Econ	16000
Brown	400	CS	15000
Smith	500	Fin	18000

DEPART

Dept	MainOffice	Phone
CS	404	9800
Econ	200	9811
Fin	501	9822
Hist	100	9833

a. Find all the information on every employee including regarding their department info.

$$\text{EMP} \bowtie_{\text{EMP.Dept = DEPART.Dept}} \text{DEPART}$$

Name	Office	EMP.Dept	Salary	DEPART.Dept	MainOffice	Phone
Smith	400	CS	10000	CS	404	9800
Jones	220	Econ	15000	Econ	200	9811
Green	100	Econ	16000	Econ	200	9811
Brown	400	CS	15000	CS	404	9800
Smith	500	Fin	18000	Fin	501	9822

b. Find all the information on every employee including their department info where employee works in an office numbered less than department main office.

$$\text{EMP} \bowtie_{(\text{EMP.Dept=DEPART.Dept}) \wedge (\text{EMP.Office} < \text{DEPART.MainOffice})} \text{DEPART}$$

Name	Office	EMP.Dept	Salary	DEPART.Dept	MainOffice	Phone
Smith	400	CS	10000	CS	404	9800
Green	100	Econ	16000	Econ	200	9811
Brown	400	CS	15000	CS	404	9800
Smith	500	Fin	18000	Fin	501	9822

B. Natural Join $\bowtie$

In Equi-join, any attributes in common such as DEPT in above relations are repeated.

The natural join will remove such duplicate attributes. Natural join does not use any comparison operator.

Example: EMP $\bowtie$ DEPART

Name	Office	Dept	Salary	Main Office	Phone
Smith	400	CS	10000	404	9800
Jones	220	Econ	15000	200	9811
Green	100	Econ	16000	200	9811
Brown	400	CS	15000	404	9800
Smith	500	Fin	18000	501	9822

C. Outer Join

The *outer* join is an extension of the *inner* join.

Outer joins of two or more tables perform an inner join of those tables according to a specified join condition and also return rows from the left join table, the right join table, or both, that do not match the inner join condition, extending the results rows with nulls in the non matching fields.

i. Left Outer Join ⟕

It includes all tuples from left hand relation and include only those matching tuples from right hand relation.

ii. Right Outer Join ⟖

It includes all tuples from right hand relation and include only those matching tuples from left hand relation.

iii. Full Outer Join ⟗

It includes all tuples from both left hand relation and right hand relation.

Examples: Assume we have two relations PEOPLE and MENU

Menu

Food	Day
Pizza	Monday
Hamburger	Tuesday
Chicken	Wednesday
Pasta	Thursday
Tacos	Friday

People

Name	Age	Food
Arbin	21	Hamburger
Bipin	24	Pizza
Chitra	23	Beer
Dines	19	Shrimp

a. **PEOPLE⟕**$_{\textbf{PEOPLE.FOOD = MENU.Food}}$ **MENU**

Name	Age	PEOPLE.Food	MENU.Food	Day
Arbin	21	Hamburger	Hamburger	Tuesday
Bipin	24	Pizza	Pizza	Monday
Chitra	23	Beer	NULL	NULL
Dines	19	Shrimp	NULL	NULL

b. **PEOPLE⟖**$_{\textbf{PEOPLE.Food = MENU.Food}}$ **MENU**

Name	Age	PEOPLE.Food	MENU.Food	Day
Bipin	24	Pizza	Pizza	Monday
Arbin	21	Hamburger	Hamburger	Tuesday
NULL	NULL	NULL	Chicken	Wednesday
NULL	NULL	NULL	Pasta	Thursday
NULL	NULL	NULL	Tacos	Friday

c. PEOPLE ⟗$_{\text{PEOPLE.Food = MENU.Food}}$ MENU

Name	Age	PEOPLE.Food	MENU.Food	Day
Arbin	21	Hamburger	Hamburger	Tuesday
Bipin	24	Pizza	Pizza	Monday
Chitra	23	Beer	NULL	NULL
Dines	19	Shrimp	NULL	NULL
NULL	NULL	NULL	Chicken	Wednesday
NULL	NULL	NULL	Pasta	Thursday
NULL	NULL	NULL	Tacos	Friday

10. Aggregate Function

The aggregate functions summarizes the values for a column. There are five aggregate function which are sum(), avg(), count(), min(), and max(). In relational algebra the aggregate function are written as below

$$G_{\text{aggregate function(attribute)}}\ R$$

Example: let us assume the following STUDENT relation

STUDENT

Name	Maths	Science
Ram	77	99
Shyam	88	44
Hari	66	55

i. Find the average marks in Maths for all the student.

$$G_{\text{avg(Maths)}}\ \text{STUDENT}$$

77

ii. Find the highest mark obtained in Science

$$G_{\text{max(Science)}}\ \text{STUDENT}$$

99

iii. Display the name of student who have scored minimum mark in Science

$$\prod_{Name}(\sigma_{Science = Gmin(Science)}(\textbf{student}))$$

Name
Shyam

Modification of database using relational algebra

The content of database can be modified by using the following operations

1. Deletion

   A delete request is expressed similarly to a query, except that instead of displaying tuples to the user, it remove the selected tuples from database. A delete operation deletes the whole tuple not a value of particular attribute/column. A deletion is expressed in relational algebra by $r \rightarrow r - E$ where r is a relation and E is a relational algebra query.

   Example: In the above relation STUDENT , Delete the record of Ram.

   $$\textbf{Student} \rightarrow \textbf{Student} - \sigma_{Name='Ram'}(\textbf{Student})$$

2. Insertion

   In relation algebra, an insertion is expressed by $r \rightarrow r \cup E$ where r is a relation and E is a relational algebra expression.

   Example: In the above relation STUDENT , Insert information of student Hari who has scored 77 in Maths and 88 in Science

   $$\textbf{Student} \rightarrow \textbf{Student} \cup \{\text{'Hari'},77,88\}$$

3. Updating

   An updation is expressed in relational algebra by $r \rightarrow \prod_{f1,\ f2,\ f3----}(r)$

   In the above relational algebra, Each f_i is the i^{th} attribute of r.

   Example: In the above relation STUDENT, increment the marks of Sciene by 10% for all the students

   $$\textbf{Student} \rightarrow \prod_{Name,Maths,Science*1.1}(\textbf{Student})$$

View

Any relation that is not part of the logical model but it is made visible to user as a virtual relation is called view.

A view is a virtual table i.e. a view looks like a table and acts like a table as far as a user is concerned. View doesn’t require physical storage.

Advantages:

1. View can be used to create logical column.
2. View can be used to maintain the summarized data by using various aggregate functions.
3. Complexity can be avoided
4. View can join multiple tables and display into a single virtual table.

View can be classified into two types

1. Simple View: I f the view is created from a single table, it is called a simple view.
2. Complex View: If the view is created on multiple table, it is called complex view.

The Syntax for creating view is

Create view <View Name> as <Query Expression>

Q. Create View for Displaying all the records.

RA: Create View V_Student1 as $\prod_{\textbf{Name,Maths,Science}}$**(STUDENT)**

Assignment:

1. Define Constraints in Relational Model and Key Integrity. What is referential Integrity?

2. Write the difference between Relational Algebra and Relational Calculus.

3. How inner join different from outer join? Point out the differences.

4. How Relational Algebra is different from Relational Calculus? Define TRC and DRC.

3. All the Questions from this chapter in Pokhara University.