Dbms unit 2 - Dbms Unit 2 notes!!

B.Tech CSE 3rd year (unit one) DBMS (Dr. A.P.J. Abdul Kalam Technical University)

Relational Model

Data is stored in tables having relationships between them called the Relational model. The relational model supports the operations like Data definition, Data manipulation, and Transaction management. Each column has a distinct name and they are representing attributes. Each row represents a single entity.

Advantages of the Relational Model

- **Simple model:** Relational Model is simple and easy to use in comparison to other languages.
- **Flexible:** Relational Model is more flexible than any other relational model present.
- **Secure:** Relational Model is more secure than any other relational model.
- **Data Accuracy:** Data is more accurate in the relational data model.
- **Data Integrity:** The integrity of the data is maintained in the relational model.
- **Operations can be Applied Easily:** It is better to perform operations in the relational model.

Disadvantages of the Relational Model

- Relational Database Model is not very good for large databases.
- Sometimes, it becomes difficult to find the relation between tables.
- Because of the complex structure, the response time for queries is high.

Characteristics of the Relational Model

- Data is represented in rows and columns called relations.
- Data is stored in tables having relationships between them called the Relational model.
- The relational model supports the operations like Data definition, Data manipulation, and Transaction management.
- Each column has a distinct name and they are representing attributes.
- Each row represents a single entity.

What is the Relational Model?

The relational model represents how data is stored in Relational Databases. A relational database consists of a collection of tables, each of

which is assigned a unique name. Consider a relation STUDENT with attributes ROLL_NO, NAME, ADDRESS, PHONE, and AGE shown in the table.

Table Student

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI		18

Important Terminologies

- **Attribute:** Attributes are the properties that define an entity. e.g.; **ROLL_NO**, **NAME, ADDRESS**
- **Relation Schema:** A relation schema defines the structure of the relation and represents the name of the relation with its attributes. e.g.; STUDENT (ROLL_NO, NAME, ADDRESS, PHONE, and AGE) is the relation schema for STUDENT. If a schema has more than 1 relation, it is called Relational Schema.
- **Tuple:** Each row in the relation is known as a tuple. The above relation contains 4 tuples, one of which is shown as:

1	RAM	DELHI	9455123451	18

- **Relation Instance:** The set of tuples of a relation at a particular instance of time is called a relation instance. Table 1 shows the relation instance of STUDENT at a particular time. It can change whenever there is an insertion, deletion, or update in the database.
- **Degree:** The number of attributes in the relation is known as the degree of the relation. The **STUDENT** relation defined above has degree 5.

- **Cardinality:** The number of tuples in a relation is known as cardinality. The **STUDENT** relation defined above has cardinality 4.
- **Column:** The column represents the set of values for a particular attribute. The column **ROLL_NO** is extracted from the relation STUDENT.

ROLL_NO
1
2
3
4

- **NULL Values:** The value which is not known or unavailable is called a NULL value. It is represented by blank space. e.g.; PHONE of STUDENT having ROLL_NO 4 is NULL.
- **Relation Key:** These are basically the keys that are used to identify the rows uniquely or also help in identifying tables. These are of the following types.
    - Primary Key
    - Candidate Key
    - Super Key
    - Foreign Key
    - Alternate Key
    - Composite Key

Constraints in Relational Model

While designing the Relational Model, we define some conditions which must hold for data present in the database are called Constraints. These constraints are checked before performing any operation (insertion, deletion, and updation) in the database. If there is a violation of any of the constraints, the operation will fail.

Domain Constraints
These are attribute-level constraints. An attribute can only take values that lie inside the domain range. e.g.; If a constraint AGE>0 is applied to STUDENT relation, inserting a negative value of AGE will result in failure.

Key Integrity
Every relation in the database should have at least one set of attributes that defines a tuple uniquely. Those set of attributes is called keys. e.g.; ROLL_NO in STUDENT is key. No two students can have the same roll number. So a key has two properties:

- It should be unique for all tuples.
- It can't have NULL values.

Referential Integrity
When one attribute of a relation can only take values from another attribute of the same relation or any other relation, it is called referential integrity. Let us suppose we have 2 relations

Table Student

ROLL_NO	NAME	ADDRESS	PHONE	AGE	BRANCH_CODE
1	RAM	DELHI	9455123451	18	CS
2	RAMESH	GURGAON	9652431543	18	CS
3	SUJIT	ROHTAK	9156253131	20	ECE
4	SURESH	DELHI		18	IT

Table Branch

BRANCH_CODE	BRANCH_NAME
CS	COMPUTER SCIENCE

BRANCH_CODE	BRANCH_NAME
IT	INFORMATION TECHNOLOGY
ECE	ELECTRONICS AND COMMUNICATION ENGINEERING
CV	CIVIL ENGINEERING

BRANCH_CODE of STUDENT can only take the values which are present in BRANCH_CODE of BRANCH which is called referential integrity constraint. The relation which is referencing another relation is called REFERENCING RELATION (STUDENT in this case) and the relation to which other relations refer is called REFERENCED RELATION (BRANCH in this case).

Anomalies in the Relational Model

An anomaly is an irregularity or something which deviates from the expected or normal state. When designing databases, we identify three types of anomalies: Insert, Update, and Delete.

Insertion Anomaly in Referencing Relation

We can't insert a row in REFERENCING RELATION if referencing attribute's value is not present in the referenced attribute value. e.g.; Insertion of a student with BRANCH_CODE 'ME' in STUDENT relation will result in an error because 'ME' is not present in BRANCH_CODE of BRANCH.

Deletion/ Updation Anomaly in Referenced Relation:

We can't delete or update a row from REFERENCED RELATION if the value of REFERENCED ATTRIBUTE is used in the value of REFERENCING ATTRIBUTE. e.g; if we try to delete a tuple from BRANCH having BRANCH_CODE 'CS', it will result in an error because 'CS' is referenced by BRANCH_CODE of STUDENT, but if we try to delete the row from BRANCH with BRANCH_CODE CV, it will be deleted as the value is not been used by referencing relation. It can be handled by the following method:

On Delete Cascade

It will delete the tuples from REFERENCING RELATION if the value used by REFERENCING ATTRIBUTE is deleted from REFERENCED

RELATION. e.g.; For, if we delete a row from BRANCH with BRANCH_CODE ‘CS’, the rows in STUDENT relation with BRANCH_CODE CS (ROLL_NO 1 and 2 in this case) will be deleted.

On Update Cascade

It will update the REFERENCING ATTRIBUTE in REFERENCING RELATION if the attribute value used by REFERENCING ATTRIBUTE is updated in REFERENCED RELATION. e.g;, if we update a row from BRANCH with BRANCH_CODE ‘CS’ to ‘CSE’, the rows in STUDENT relation with BRANCH_CODE CS (ROLL_NO 1 and 2 in this case) will be updated with BRANCH_CODE ‘CSE’.

Super Keys

Any set of attributes that allows us to identify unique rows (tuples) in a given relationship is known as super keys. Out of these super keys, we can always choose a proper subset among these that can be used as a primary key. Such keys are known as Candidate keys. If there is a combination of two or more attributes that are being used as the primary key then we call it a Composite key.

Entity integrity

Entity integrity is concerned with ensuring that each row of a table has a unique and non-null primary key value; this is the same as saying that each row in a table represents a single instance of the entity type modelled by the table.

- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

Example:

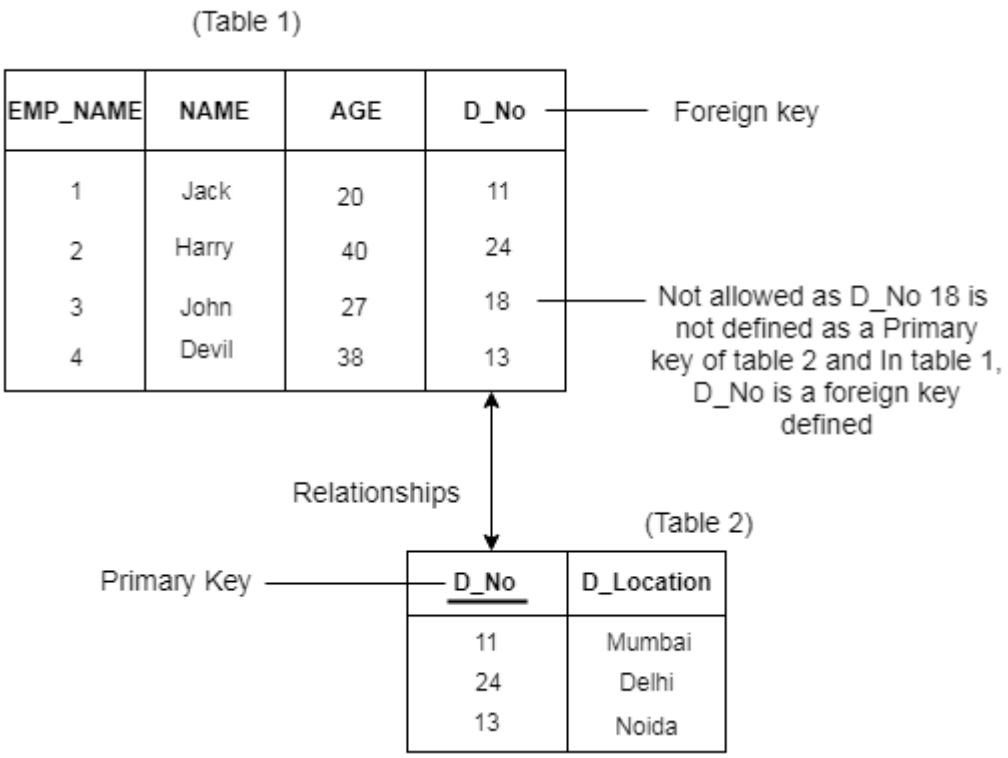
EMPLOYEE

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.



Key constraints

- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

Not allowed. Because all row must be unique

Relational algebra

Relational algebra refers to a procedural query language that takes relation instances as input and returns relation instances as output. It performs queries with the help of operators. A binary or unary operator can be used. They take in relations as input and produce relations as output.

Relational Algebra is a procedural query language. Relational algebra mainly provides a theoretical foundation for relational databases and SQL. The main purpose of using Relational Algebra is to define operators that transform one or more input relations into an output relation. Given that these operators accept relations as input and produce relations as output, they can be combined and used to express potentially complex queries that transform potentially many input relations (whose data are stored in the database) into a single output relation (the query results). As it is pure mathematics, there is no use of English Keywords in Relational Algebra and operators are represented using symbols.

Fundamental Operators

These are the basic/fundamental operators used in Relational Algebra.

1. Selection(σ)
2. Projection(π)
3. Union(U)
4. Set Difference(-)
5. Set Intersection(∩)
6. Rename(ρ)
7. Cartesian Product(X)

1. Selection(σ): It is used to select required tuples of the relations.

A	B	C
1	2	4
2	2	3
3	2	3

A	B	C
4	3	4

For the above relation, **σ(c>3)R** will select the tuples which have c more than 3.

A	B	C
1	2	4
4	3	4

Note: The selection operator only selects the required tuples but does not display them. For display, the data projection operator is used.

it selects tuples that satisfy the given predicate from a relation.

Notation – $\sigma_p(r)$

Where **σ** stands for selection predicate and **r** stands for relation. *p* is prepositional logic formula which may use connectors like **and, or,** and **not**. These terms may use relational operators like – =, ≠, ≥, < , >, ≤.

For example –

$\sigma_{subject = "database"}(Books)$

Output – Selects tuples from books where subject is 'database'.

$\sigma_{subject = "database" \text{ and } price = "450"}(Books)$

Output – Selects tuples from books where subject is 'database' and 'price' is 450.

$\sigma_{subject = "database" \text{ and } price = "450" \text{ or } year > "2010"}(Books)$

Output – Selects tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

2. Projection(π): It is used to project required column data from a relation.
Example: Consider Table 1. Suppose we want columns B and C from Relation R.
π(B,C)R will show following columns.

B	C
2	4
2	3
3	4

Note: By Default, projection removes duplicate data.
It projects column(s) that satisfy a given predicate.

It projects column(s) that satisfy a given predicate.

Notation – $\prod_{A1, A2, An}$ (r)

Where A_1, A_2 , A_n are attribute names of relation **r**.

Duplicate rows are automatically eliminated, as relation is a set.

For example –

$\prod_{subject, author}$ (Books)
Selects and projects columns named as subject and author from the relation Books.

3. Union(U): Union operation in relational algebra is the same as union operation in set theory.
Example:

FRENCH
0 seconds of 15 secondsVolume 0%

Student_Name	Roll_Number
Ram	01
Mohan	02
Vivek	13
Geeta	17

GERMAN

Student_Name	Roll_Number
Vivek	13
Geeta	17
Shyam	21
Rohan	25

Consider the following table of Students having different optional subjects in their course.

π(Student_Name)FRENCH U π(Student_Name)GERMAN

Student_Name
Ram
Mohan
Vivek
Geeta
Shyam
Rohan

Note: The only constraint in the union of two relations is that both relations must have the same set of Attributes.

It performs binary union between two given relations and is defined as –

$r \cup s = \{ t \mid t \in r \text{ or } t \in s\}$

Notation – r U s

Where **r** and **s** are either database relations or relation result set (temporary relation).

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

- **r**, and **s** must have the same number of attributes.
- Attribute domains must be compatible.

- Duplicate tuples are automatically eliminated.

$\prod_{author}$ (Books) $\cup \prod_{author}$ (Articles)

Output – Projects the names of the authors who have either written a book or an article or both.

4. Set Difference(-): Set Difference in relational algebra is the same set difference operation as in set theory.
Example: From the above table of FRENCH and GERMAN, Set Difference is used as follows
π(Student_Name)FRENCH - π(Student_Name)GERMAN

Student_Name
Ram
Mohan

Note: The only constraint in the Set Difference between two relations is that both relations must have the same set of Attributes.

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation – **r** – **s**

Finds all the tuples that are present in **r** but not in **s**.

$\prod_{author}$ (Books) – $\prod_{author}$ (Articles)

Output – Provides the name of authors who have written books but not articles.

5. Set Intersection(∩): Set Intersection in relational algebra is the same set intersection operation in set theory.

Example: From the above table of FRENCH and GERMAN, the Set Intersection is used as follows

π(Student_Name)FRENCH ∩ π(Student_Name)GERMAN

Student_Name
Vivek
Geeta

Note: The only constraint in the Set Difference between two relations is that both relations must have the same set of Attributes.

6. Rename(ρ): Rename is a unary operation used for renaming attributes of a relation.

ρ(a/b)R will rename the attribute 'b' of the relation by 'a'.

7. Cross Product(X): Cross-product between two relations. Let's say A and B, so the cross product between A X B will result in all the attributes of A followed by each attribute of B. Each record of A will pair with every record of B.

Example:

A

Name	Age	Sex
Ram	14	M
Sona	15	F
Kim	20	M

B

ID	Course
1	DS
2	DBMS

A X B

Name	Age	Sex	ID	Course
Ram	14	M	1	DS
Ram	14	M	2	DBMS
Sona	15	F	1	DS
Sona	15	F	2	DBMS
Kim	20	M	1	DS
Kim	20	M	2	DBMS

Note: If A has ‘n’ tuples and B has ‘m’ tuples then A X B will have ‘ n*m ‘ tuples.

Combines information of two different relations into one.

Notation − r X s

Where **r** and **s** are relations and their output will be defined as −

r X s = { q t | q ∈ r and t ∈ s}

$\sigma_{author = 'tutorialspoint'}$(Books X Articles)

Output – Yields a relation, which shows all the books and articles written by tutorialspoint.

Derived Operators

These are some of the derived operators, which are derived from the fundamental operators.

1. Natural Join(⋈)
2. Conditional Join

1. Natural Join(⋈): Natural join is a binary operator. Natural join between two or more relations will result in a set of all combinations of tuples where they have an equal common attribute.

Example:

EMP

Name	ID	Dept_Name
A	120	IT
B	125	HR
C	110	Sales
D	111	IT

DEPT

Dept_Name	Manager
Sales	Y
Production	Z

Dept_Name	Manager
IT	A

Natural join between EMP and DEPT with condition :

EMP.Dept_Name = DEPT.Dept_Name

EMP ⋈

DEPT

Name	ID	Dept_Name	Manager
A	120	IT	A
C	110	Sales	Y
D	111	IT	A

2. Conditional Join: Conditional join works similarly to natural join. In natural join, by default condition is equal between common attributes while in conditional join we can specify any condition such as greater than, less than, or not equal.

Example:

R

ID	Sex	Marks
1	F	45
2	F	55
3	F	60

ID	Sex	Marks

S

ID	Sex	Marks
10	M	20
11	M	22
12	M	59

Join between R and S with condition **R.marks >= S.marks**

R.ID	R.Sex	R.Marks	S.ID	S.Sex	S.Marks
1	F	45	10	M	20
1	F	45	11	M	22
2	F	55	10	M	20
2	F	55	11	M	22
3	F	60	10	M	20
3	F	60	11	M	22

R.ID	R.Sex	R.Marks	S.ID	S.Sex	S.Marks
3	F	60	12	M	59

Relational Calculus

As Relational Algebra is a procedural query language, Relational Calculus is a non-procedural query language. It basically deals with the end results. It always tells me what to do but never tells me how to do it.
There are two types of Relational Calculus

1. Tuple Relational Calculus(TRC)
2. Domain Relational Calculus(DRC)