DataBase Management System (DBMS).

Date
Page 1.

UNIT-II

Relational Query Languages

☺ Relational Algebra.

→ Relational Algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output.

→ It uses operators to perform queries.
An operator can be either: Unary or Binary.

→ They accept relations as their input and yield relations as their output.

The fundamental operations of relational Algebra are as follows-

1) SELECT
2) PROJECT
3) UNION
4) SET DIFFERENT
5) CARTESIAN PRODUCT
6) RENAME.

1) **Select operation (σ)** It selects tuples that satisfy the given predicate from a relation.

Notation - $\sigma_p(r)$

where,

σ stands for Selection predicate
r stands for relation.
p is propositional logic formula (which may use connectors like and, or and not.

Example :- $\sigma_{subject = "database"}(BOOKS)$

Output :- Select tuples from books where subject is 'database'.

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

2) **Project Operation (Π)** It projects column(s) that satisfy a given predicate.

Notation: $\Pi_{A1, A2, An}(r)$

where, A1, A2, An are attribute names of relation(r).
Duplicate rows are automatically eliminate, as relation is a set.

Example :- $\Pi_{subject, author}(BOOKS)$
selects and projects columns named as Subject and author from the relation Books.

3) **Union Operation (∪)** It performs binary union between two given relation and is defined as:-

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

Notation: $r \cup s$

where,

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

The following conditions must be hold:-

1) r and s have the same number of attributes
2) Attributes domains must be compatible.
3) Duplicates tuples are automatically eliminated.

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

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

4) **Set difference:-** The result of set difference operation is tuples which are present in one ~~operation~~ relation but are not in the second relation.

Notation: r - s

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

$\Pi_{author}(Books) - \Pi_{author}(Articles)$

Output :- provides the name of authors who have written books but not articles.

5) Cartesian Product (X) :- Combines information of two different relations into one.

Notation :- r X s

where r and s are relations

$r \times s = \{ q\ t \mid q \in r \text{ and } t \in s \}$

$Q_{author} = \text{'srishti'}(Books) \times Articles)$.

Output :- yields a relation, which shows all the books and articles written by srishti.

6) Rename Operation (ρ) The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation.

Notation :- $\rho_x(E)$ [ρ - rho]

where the result of expression (E) is saved with name of x.

Additional operations are :-

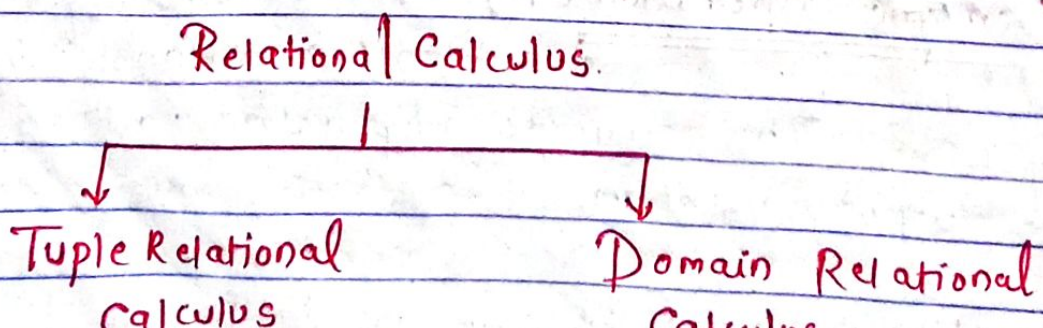
1) Set Intersection
2) Assignment
3) Natural join

☺ Tuple and Domain Relational Calculus.

→ Relational Calculus :-

(a) 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.

Types of Relational Calculus



3. Tuple Relational Calculus (TRC)

→ The tuple relation calculus is specified to select the tuples in a relation.
In TRC,
filtering variables uses the tuples of a relation.

→ The results of the relation can have one or more tuples.

Notation:- $\{T \mid P(T)\}$ or $\{T \mid Condition(T)\}$
Where,
T is resulting Tuples.
P(T) is the condition used to fetch T.

Example: $\{T.name \mid Author(T) \text{ AND } T.article = 'database'\}$

OUTPUT:- This query selects the tuples from the AUTHOR relation. It returns a tuple with 'name' from Author who has written an article on 'database'.

TRC (tuple relation calculus) can be quantified. In TRC we use Existential ($\exists$) and Universal Quantifiers ($\forall$)

Ex: $\{R \mid \exists T \in Authors(T.article = 'database' \text{ AND } R.name = T.name)\}$

Output: This query will yield the same result as the previous one.

2. Domain Relational Calculus. (DRC).

→ The second form of relation is known as Domain Relation 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 ∧ (and), ∨ (or) and ¬ (not).

→ It uses Existential (∃) and Universal Quantifiers (∀) to bind the variable.

Notation: $\{ a_1, a_2, a_3 \ldots a_n \mid P(a_1, a_2, a_3 \ldots a_n) \}$

Where

a_1, a_2 are attributs

P stands for formula built by inner attributes.

Example:-

$\{ \langle article, page, subject \rangle \mid \in books \wedge subject = \text{"database"} \}$

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

SQL3

{ programming language }

SQL: 1999 (also called SQL3) was the fourth revision of the SQL database query language.

ABOUT SQL3

→ The new SQL3 datatypes gives a relational database more flexibility in what can be used as a type for a table column.

→ SQL3 includes data defination, and management techniques from Object-Oriented dbms, OO-dbms while maintaining the relation dbms platform.

New features in SQL3

→ Classification hierarchies

→ Embedded structures that support composite attributes.

→ Collection data-types (sets, lists / arrays and multi-sets) that can be used for multi-valued attribute. types.

→ Large Object types, LOBs, within the DB, as opposed to requiring external storage.

→ Users defined data-types and functions (UDT/UDF) that can be defined complex structures and derived attributes value calculation among many funt^n ext.

DDL and DML Constructs

SQL Commands.

DDL	DML	DCL	TCL
Data Definition lang.	Data Manipulation lang.	Data Control	Transcation Control
Create	Select	Grant	Commit
Alter	Insert	Revoke	Rollback
Drop	Update		Savepoint
Truncate	Delete		Set Transcation
Comment	Merge		
Rename	Call		
	Explain Plan		
	Lock Table		

	Key	DDL	DML
1.	Stands for	DDL stands for Data Defination language.	DML stands for Data manipulation language.
2.	Usage.	DDL statements are used to create database schema, constraints, Users, tables etc.	DML statement is used to insert, update or delete the records.

Classification	DDL has no further classification	DML is further classified into procedural DML and non-procedural DML.
Commands	CREATE, DROP, RENAME and ALTER	INSERT, UPDATE. and DELETE.

(9) Open Source and Commercial DBMS.

1. **Open Source Database:** It is database that anyone can easily view the source code and this is open and free to download. Also for community version some small additional and affordable cost are imposed.

Open Source Database provide limited technical support to end users.

Here Installation and Updates are administered by users.

Examples:- MYSQL, PostgreSQL, MongoDB etc.

2. **Commercial Database:** Commercial database are that which has been created for commercial purpose only.
They are premium and are not free like Open Source Database. In commercial Database it is guaranteed that tecnnical support is provided

In this Installation and updates are Administrated by software Vendor.

Example:- Oracle, IBM DB2 etc.

Relational Database Design

(models information and data into a set of tables with rows and columns).

Domain and Data dependency:-

Dependency:- In DBMS is a relation between two or more attributes.

It has following types in DBMS:-

1) functional Dependency
2) fully-functional Dependency
3) Transitive Dependency
4) Multivalued Dependency
5) Partial Dependency

Functional Dependency :-

→ Functional dependency in DBMS as the name suggests it is a relationship between attributes of a table dependent on each other.

→ It helps in preventing data redundancy and gets to know about bad designs.

Let us consider,

P is a relation with Attributes A and B

→ Functional Dependency is represented by → (arrow sign).

Ex:- $A \rightarrow B$

Functional Dependency

$A \rightarrow B$

B - functionally dependent on A

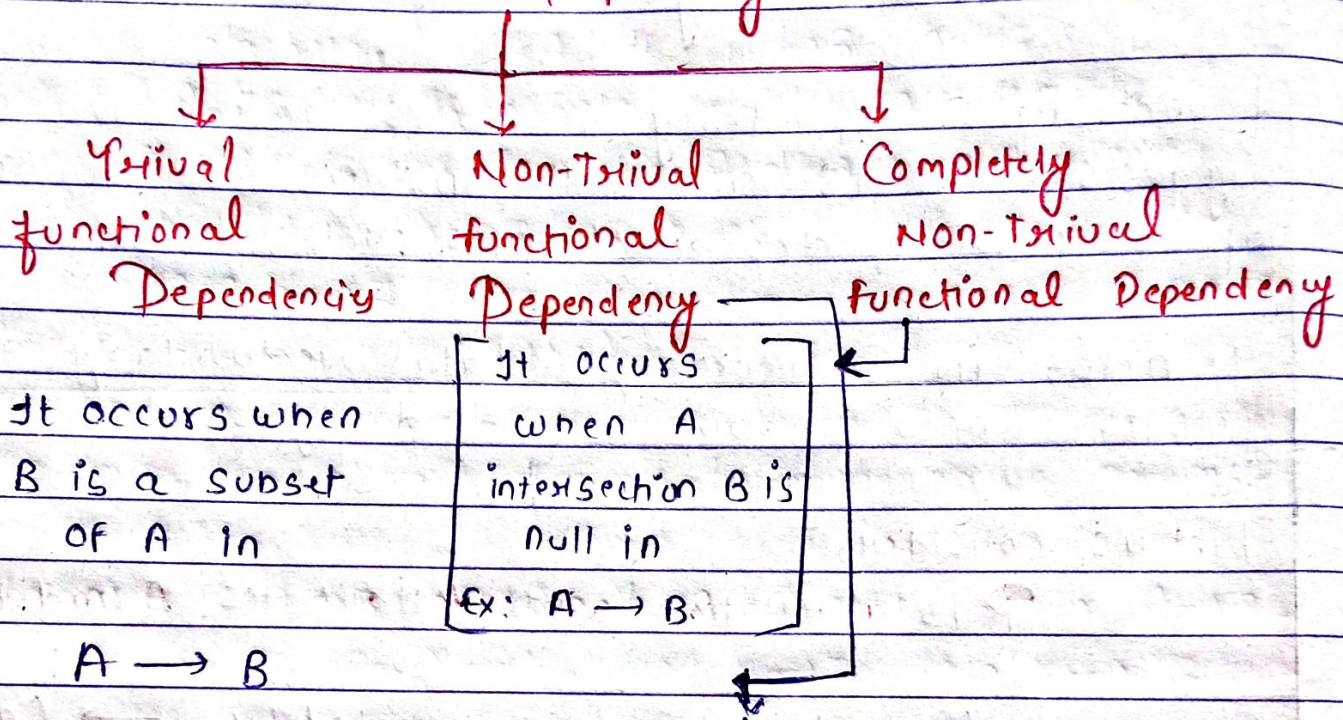
A - determinant set

B - dependent Attribute

Example :- DeptId → DeptName

Here, DeptId and deptName can be determined as DeptId is functionally dependent on DeptName.

Functional Dependency



Armstrong's Axioms.

Introduced by:- William W. Armstrong

(i) Armstrong refers to the sound and complete set of inference rules or axioms, introduced by William Armstrong that is used to test the logical implication of functional Dependencies.

If f is a set of functional dependencies then the closure of F, denoted by F^+, is all set of all functional dependencies logically implied by F.

Axioms

1. Axiom of Reflexivity :- If A is a set of attributes and B is subset of C, then C holds B
   If B ⊂ A then A → B
   This property of trivial property.

2. Axiom of Augmentation:- If A → B holds and Y is attributes set, then AY → BY also holds. That is adding attributes in dependencies, does not change the basic dependencies. If A → B, then AC → BC for any C.

3. **Axiom of transitivity:-** Same as the transitive rule in algebra, if $A \rightarrow B$ holds and $B \rightarrow C$ holds, then $A \rightarrow C$ also holds. $A \rightarrow B$ is called as A functionally that determines B. If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$.

☺ **Secondary Rules:-**

These rules can be derived from the above axioms.

1. **Union:-** If $A \rightarrow B$ holds and $A \rightarrow C$ holds, then $A \rightarrow BC$ holds. If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$.

2. **Composition:-** If $A \rightarrow B$ and $X \rightarrow Y$ holds, then $AX \rightarrow BY$

3. **Decomposition:-** If $A \rightarrow BC$ holds then $A \rightarrow B$ and $A \rightarrow C$ hold. If $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$.

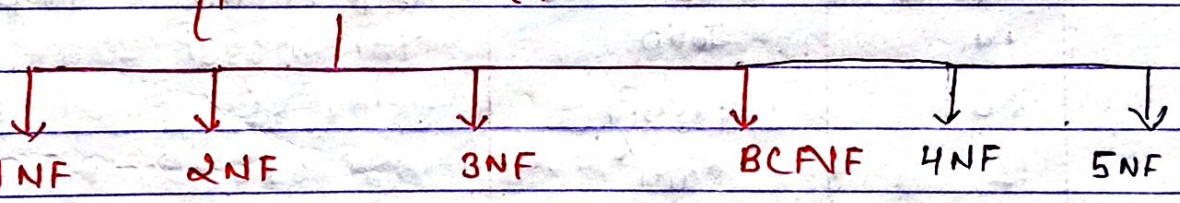
4. **Pseudo Transivity:-** If $A \rightarrow B$ holds and $BC \rightarrow D$ holds, then $AC \rightarrow D$ holds. If $X \rightarrow Y$ and $YZ \rightarrow W$ then $XZ \rightarrow W$.

NORMAL FORMS

DBMS Normalization :-

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations.
- It is used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization divides the larger table into the smaller table and links them using relationship.
- The normal form is used to reduce redundancy from the database table.

Types of Normal Forms



1. 1NF :- A relation is in 1NF if it contains an atomic value.
2. 2NF :- A relation will be 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3. 3NF :- A relation will be 3NF if it is in 2NF and no transition dependency exists.
4. 4NF :- A relation will be 4NF if it is Boyce codd normal

and has no multi-valued dependency.

(5) 5NF :- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

1) First Normal form :- (1NF)

→ A relation will be 1NF if it contains an atomic value.

→ It states that an attribute of a table cannot hold multiple values. It must only single-valued attribute.

→ first normal forms disallows the multi-valued attribute, composite attribute and their combinations.

Example :- Relation Employee is not in 1NF because of multi-valued attribute Emp-Phone.

Employee table

Emp-ID	Emp-Name	Emp-Phone	Emp-state
14	John	7272826385	UP
20	Harry	8574783832	Bihar
12	Sam	9952638210	Punjab.

2) Second Normal form :- (2NF)

In the 2NF, relational must be in 1NF.

In the second normal form, all non-key attributes are fully functional dependent on the primary key.

Example: Let's assume, a school can store the data of teachers and the subjects they teach.
In a school, a teacher can teach more than one subject

Teacher Table

Teacher_ID	Subject	Teacher_age
25	chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

In the given Table,
Teacher_age Non-prime attribute
is dependent on Teacher_id
which is a proper subset of a Candidate key.
That's why it violates the rule for 2NF.
To convert the given table into 2NF, we decompose it into two tables.

Teacher-detail table:-

Teacher_id	Teacher_age
25	30
47	35
83	38

Teacher_subject Table.

Teacher_id	subject
25	chemistry
25	Biology
47	English
83	Math
83	computer.

3) Third Normal Form (3NF) :-

- A relation will be in 3NF if it is 2NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication.
- If there is no transitive dependency for non-prime attributes. then the relation must be in third normal form.

(v) Conditions for every non-trivial function dependency $X \to Y$

1. X is a super key
2. Y is a prime attribute i.e. each element of Y. is part of some candidate key.

Example :-

Emp Id	Emp-Name	Emp-zip	Emp-state	Emp-city
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago

Super Key : {Emp-ID}, {Emp-Id, Emp-Name} {Emp-ID, Emp-Name, Emp-zip ... so}

Candidate Key: {Emp-Id}

Non-prime Attribute: Emp-Id are non-prime attribute

Here, Emp-state & Emp city dependent on Emp-zip and Emp-zip dependent on Emp-Id

The non-prime attribute (Emp-state, Emp-city) transitively dependent on superkey (Emp-Id). It violate

the rule of third normal form.

We need to move the Emp-city and Emp-state to the new <Employee-Zip> table, Emp-Zip as a primary key.

Employee table

Emp-Id	Emp_Name	Emp-Zip
222	Harry	201010
333	Stephan	02228
444	Lan	60007

Employee-Zip table:

Emp_Zip	Emp-State	Emp-City
201010	UP	Noida
02228	US	Boston
60007	US	chicago

Boyce Codd Normal Form (BCNF)

BCNF is the advance version of 3NF. It is stricter than 3NF.

A table is in BCNF if every functional dependency $X \rightarrow Y$, X is the super key of table.

For BCNF, the table should be in 3NF, and every FD, LHS is super key.

Example:- Let us assume there is a company where employee work in more than one department.

Employee table:-

Emp-Id	Emp-country	Emp-Dept	Dept-type	Emp-Dept-no
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D284	232
364	UK	Developing	D284	549

Emp-Id → Emp-country

Emp-dept → {Dept-type, Emp-dept-no}

Candidate Key: {Emp-Id, Emp-Dept}

The table is not in BCNF because neither Emp-Dept nor Emp-Id alone are Keys.

To convert the given table,
We decompose it into three tables.

- Emp-country table:

Emp-Id	Country
264	India
264	India

- Emp-Dept table:

Emp-Dept	Dept-type	Emp Dept-no
Designing	D394	283
Testing	D394	300
Stories	D284	232
Developing	D284	549

- Emp-Dept-mapping table:-

Emp-id	Emp-Dept
D394	283
D394	300
D283	232
D283	549

Functional Dependencies:

Emp_Id → Emp-country Emp-Dept → {Dept-type, Emp-Dept-no}

Candidate Key:

1st table — Emp-Id

2nd table — Emp-dept

3rd table — {Emp-id, Emp-dept}.

Now, this is in BCNF because left side part of the functional dependencies is a key.

4) FOURTH NORMAL FORM (4NF)

- A relation will be HNF if it in Boyce Codd normal form and has no multivalued dependency.
- for a dependency A → B if for a single value of A, multiple values of B exists, then the relation will be a multivalued dependency.

(class)

Example :- Database table :- R1 (SID, SNAME)

SID	SNAME
S1	A
S2	B

Table : R2 (CID, CNAME)

CID	CNAME
C1	C
C2	D

When the CROSS product is done it resulted in multivalued dependencies:-

SID	SNAME	CID	CNAME
S1	A	C1	C
S1	A	C2	D
S2	B	C1	C
S2	B	C2	D

Multivalued Dependencies (MVD) are:-

SID →→ CID; SID →→ CNAME, SNAME →→ CNAME

☺ JOIN DEPENDENCY:- Join decomposition is further generalization of Multivalued dependencies.

→ If the join of R1 and R2 over C is equal to R then,

→ we can say that a join dependency (JD) exists.

→ Where R1 and R2 are a lossless decomposition of R.
→ A JD ⋈ {R1, R2, R3 --- Rn} are a lossless decomposition of R.
→ The * (A, B, C, D), (C, D) will be a JD of R if the join of join's attribute is equal to the relation R.
→ Here, * (R1, R2, R3) is used to indicate the Relation R1, R2, R3 and so on are a JD of R.

5) FIFTH NORMAL FORM (5NF)

- A Relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless
- 5NF is satisfied when all the tables are broken as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF)

Example

Subject	Lecturer	Semester
Computer	Anshika	Semester I
Computer	John	Semester 2
Math	John	Semester I
Math	Akash	Semester 2
Chemistry	Praveen	Semester I

We decompose into three relations P1, P2 & P3.

P1

Semester	Subject
Semester 1	Computer
Semester 1	Math
Semester 2	Chemistry
semester 2	Math.

P2

Subject	Lecturer
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

P3.

Semester	Lecturer
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash.
Semester 1	Praveen

Relational Decomposition :- When a relation in the relational model is not in the appropriate normal form then the decom. of a relation is required.

Date Page 27

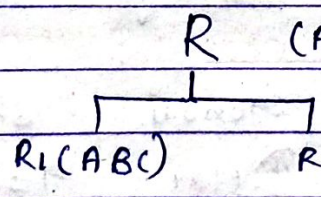
- Dependency Preserving
- Lossless Decomposition.

DEPENDENCY PRESERVATION.

- It is important constraint of the Database.
- It is the dependency preservation, at least one decomposed table must satisfy every dependency.
- If a relation R is decomposed into Relation R_1 and R_2, then the dependencies of R either must be a part of R_1 or R_2 or must be derivable from the combination of functional depencies of R_1 and R_2.

Example:- Suppose there is relation R(A, B, C, D) then, the functional dependencies of R either set(A >BC)

The Relational R is decomposed into R_1 (ABC) and R_2 (AD)



which is dependency preserving becouse FD A → BC is a part of Relation R_1 (ABC).

LOSSLESS DECOMPOSITION.

→ If the information is not lost from the relation that is decomposed, then the decomposition will be lossless.

→ The lossless decomposition guarantees that the join of relations will result in the same relation it was decomposed.

→ The relation is said to be lossless decomposition if natural joins of all the decomposition give the original ~~version~~. relation.

Example:-

Employee-Department Table:-

Emp-Id	Emp_Name	Emp-age	Emp-city	Dept-Id	Dept-name
22	Denim	28	Mumbai	827	Sales
33	Alina	25	Delhi	438	Marketing
46	Stephan	30	Bangalore	869	Finance
52	Katherine	36	Mumbai	575	Production
60	Jack	40	Noida	678	Testing

The Above relation is decomposed into two relations EMPLOYEE and DEPARTMENT.

→ Employee table:-

Emp-Id	Emp-Name	Emp-age	Emp-city
22	Denim	28	Mumbai
33	Alina	25	Delhi
46	Stephan	30	Bangalore
52	Katherine	36	Mumbai
60	Jack	40	Noida.

→ DEPARTMENT TABLE:-

Dept-Id	Emp-Id	Dept-Name.
827	22	Sales
438	33	Marketing
869	46	Finance
575	52	Production
678	60	Testing.

Now, when these two relations are joined on the common column "Emp-Id"
then, the resultant relation will look like:-
↓

Employee ⋈ Department

Emp-Id	Emp-Name	Emp-age	Emp-city	Dept-Id	Dept-name
22	Denim	28	Mumbai	827	Sales
33	Alina	25	Delhi	438	Marketing
46	Stephan	30	Bangalore	869	finance
52	Katherine	36	Mumbai	575	production
60	Jack	40	Noida	678	Testing.

Hence, the decomposition is Lossless Decomposition.