Your Dreams Our Goal
POORNIMA UNIVERSITY
Member of Association of Indian Universities & Approved by UGC (Govt. of India) under 2(f) & 12(B)
DETAILED LECTURE NOTES

Unit- 2 Lecture - 9		
Faculty of : FCE Program: BTECH	**Class/Section: IV**	**Date:**
Name of Faculty:-Mrs.Shakshi Ranawat	**Name of Course: RDBMS**	**Code: BCECCE4103**

Introduction to Distributed Database

- A distributed database is basically a database that is not limited to one system; it is spread over different sites, i.e., on multiple computers or over a network of computers.
- A distributed database system is located on various sites that don't share physical components.
- This may be required when a particular database needs to be accessed by various users globally. It needs to be managed such that for the users it looks like one single database.

Types:

1. **Homogeneous Database:**

In a homogeneous database, all different sites store database identically. The operating system, database management system, and the data structures used – all are the same at all sites. Hence, they're easy to manage.

2. **Heterogeneous Database:**

In a heterogeneous distributed database, different sites can use different schema and software that can lead to problems in query processing and transactions. Also, a particular site might be completely unaware of the other sites. Different computers may use a different operating system, different database application. They may even use different data models for the database. Hence, translations are required for different sites to communicate.

Distributed Data Storage:

There are 2 ways in which data can be stored on different sites. These are:

1. **Replication –**

In this approach, the entire relationship is stored redundantly at 2 or more sites. If the entire database is available at all sites, it is a fully redundant database. Hence, in replication, systems maintain copies of data.

This is advantageous as it increases the availability of data at different sites. Also, now query requests can be processed in parallel.

However, it has certain disadvantages as well. Data needs to be constantly updated. Any change made at one site needs to be recorded at every site that relation is stored or else it may lead to inconsistency. This is a lot of overhead. Also, concurrency control becomes way more complex as concurrent access now needs to be checked over a number of sites.

2. **Fragmentation –**

In this approach, the relations are fragmented (i.e., they're divided into smaller parts) and each of the fragments is stored in different sites where they're required. It must be made sure that the fragments are such that they can be used to reconstruct the original relation (i.e, there isn't any loss of data).

Fragmentation is advantageous as it doesn't create copies of data, consistency is not a problem.

Fragmentation of relations can be done in two ways:

Horizontal fragmentation – Splitting by rows –

The relation is fragmented into groups of tuples so that each tuple is assigned to at least one fragment.

Vertical fragmentation – Splitting by columns –

The schema of the relation is divided into smaller schemas. Each fragment must contain a common candidate key so as to ensure a lossless join.

In certain cases, an approach that is hybrid of fragmentation and replication is used.

Applications of Distributed Database:

It is used in Corporate Management Information System.

It is used in multimedia applications.

Used in Military's control system, Hotel chains etc.

It is also used in manufacturing control system.

A distributed database system is a type of database management system that stores data across multiple computers or sites that are connected by a network. In a distributed database system, each site has its own database, and the databases are connected to each other to form a single, integrated system.

The main advantage of a distributed database system is that it can provide higher availability and reliability than a centralized database system. Because the data is stored across multiple sites, the system can continue to function even if one or more sites fail. In addition, a distributed database system can provide better performance by distributing the data and processing load across multiple sites.

There are several different architectures for distributed database systems, including:

- **Client-server architecture:** In this architecture, clients connect to a central server, which manages the distributed database system. The server is responsible for coordinating transactions, managing data storage, and providing access control.
- **Peer-to-peer architecture:** In this architecture, each site in the distributed database system is connected to all other sites. Each site is responsible for managing its own data and coordinating transactions with other sites.
- **Federated architecture:** In this architecture, each site in the distributed database system maintains its own independent database, but the databases are integrated through a middleware layer that provides a common interface for accessing and querying the data.

Distributed database systems can be used in a variety of applications, including e-commerce, financial services, and telecommunications. However, designing and managing a distributed database system can be complex and requires careful consideration of factors such as data distribution, replication, and consistency.

Advantages of Distributed Database System:

1) There is fast data processing as several sites participate in request processing.

2) Reliability and availability of this system is high.

3) It possesses reduced operating cost.

4) It is easier to expand the system by adding more sites.

5) It has improved sharing ability and local autonomy.

Disadvantages of Distributed Database System:

1) The system becomes complex to manage and control.

2) The security issues must be carefully managed.

3) The system require deadlock handling during the transaction processing otherwise the entire system may be in inconsistent state.

4) There is need of some standardization for processing of distributed database system.

Your Dreams Our Goal
POORNIMA UNIVERSITY
Member of Association of Indian Universities & Approved by UGC (Govt. of India) under 2(f) & 12(B)
DETAILED LECTURE NOTES

Lecture – 10		
Faculty of : FCE Program: BTECH	**Class/Section: IV**	**Date:**
Name of Faculty:-Mrs.Shakshi Ranawat	**Name of Course: RDBMS**	**Code: BCECCE4103**

Classification of DBMS

DBMS is a collection of programs used for managing data and simultaneously it supports different types of users to create, manage, retrieve, update and store information.

Types of DBMS

The types of DBMS based on data model are as follows −

- Relational database.
- Object oriented database.
- Hierarchical database.
- Network database.

Relation Database

A **relational database management system** (RDBMS) is a system where data is organized in two-dimensional tables using rows and columns.

This is one of the most popular data models which is used in industries. It is based on **SQL**.

Every table in a database has a key field which uniquely identifies each record.

This type of system is the most widely used DBMS.

Relational database management system software is available for personal computers, workstation and large mainframe systems.

For example − Oracle Database, MySQL, Microsoft SQL Server etc.

Std ID	**Name**	**City**
201	Bob	Hyderabad
204	Lucky	Chennai
205	Pinky	Bangalore

In the above student table Std ID, Name and city are called as attributes and their values. Std ID is a primary key attribute which uniquely identifies each record in the student table.

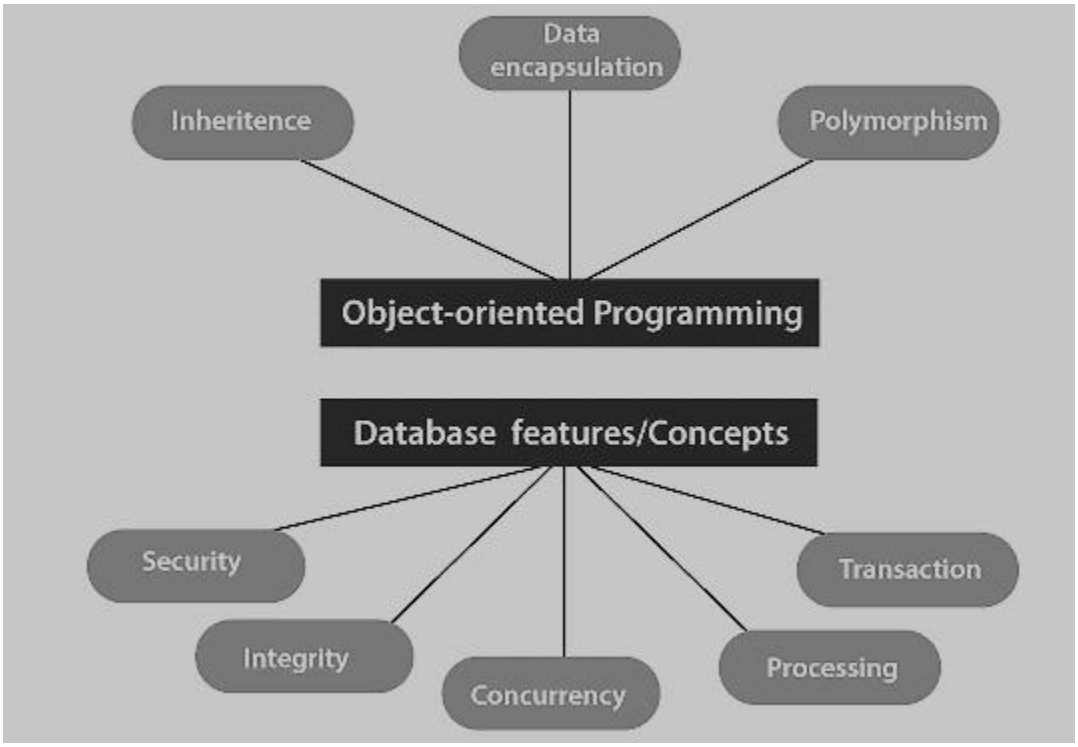
Object Oriented Database

It is a system where information or data is represented in the form of objects which is used in object-oriented programming.

- It is a combination of relational database concepts and object-oriented principles.
- Relational database concepts are concurrency control, transactions, etc.
- **OOPs principles** are data encapsulation, inheritance, and polymorphism.
- It requires less code and is easy to maintain.

For example − Object DB software.

The object oriented database is represented in diagram format below −



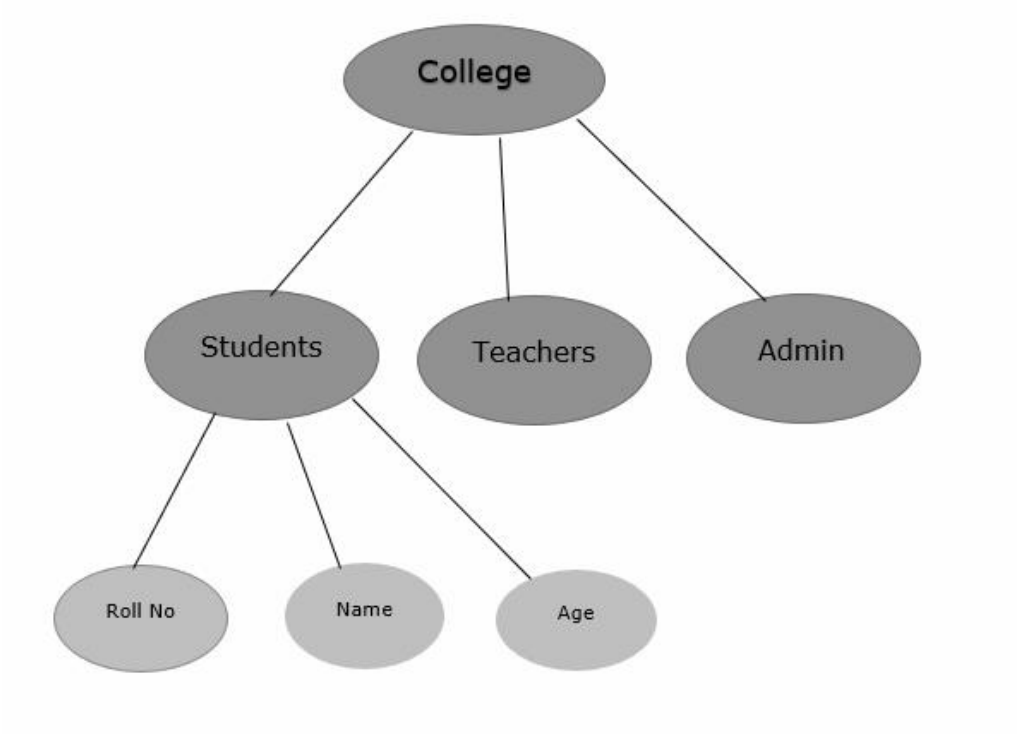
Hierarchical Database

It is a system where the data elements have a one to many relationship (1: N). Here data is organized like a tree which is similar to a folder structure in your computer system.

- The hierarchy starts from the root node, connecting all the child nodes to the parent node.
- It is used in industry on mainframe platforms.

For example− IMS (IBM), Windows registry (Microsoft).

An example of a hierarchical database is given below −



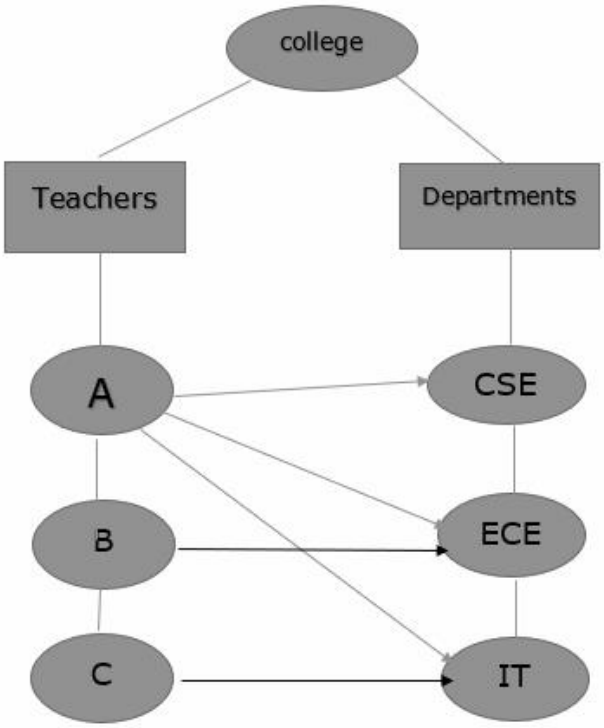
Network database

A **Network database management system** is a system where the data elements maintain one to one relationship (1: 1) or many to many relationship (N: N).

It also has a hierarchical structure, but the data is organized like a graph and it is allowed to have more than one parent for one child record.

Example

Teachers can teach in multiple departments. This is shown below −



What is RDBMS (Relational Database Management System)

RDBMS stands for *Relational Database Management System.*

All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL, and Microsoft Access are based on RDBMS.

It is called Relational Database Management System (RDBMS) because it is based on the relational model introduced by E.F. Codd.

How it works

Data is represented in terms of tuples (rows) in RDBMS.

A relational database is the most commonly used database. It contains several tables, and each table has its primary key.

Due to a collection of an organized set of tables, data can be accessed easily in RDBMS.

Brief History of RDBMS

From 1970 to 1972, E.F. Codd published a paper to propose using a relational database model.

RDBMS is originally based on E.F. Codd's relational model invention.

Characteristics of RDBMS

- Data must be stored in tabular form in DB file, that is, it should be organized in the form of rows and columns.
- Each row of table is called record/tuple . Collection of such records is known as the cardinality of the table
- Each column of the table is called an attribute/field. Collection of such columns is called the arity of the table.
- No two records of the DB table can be same. Data duplicity is therefore avoided by using a candidate key. Candidate Key is a minimum set of attributes required to identify each record uniquely.
- Tables are related to each other with the help for foreign keys.
- Database tables also allow NULL values, that is if the values of any of the element of the table are not filled or are missing, it becomes a NULL value, which is not equivalent to zero. (NOTE: Primary key cannot have a NULL value).

Advantages of RDBMS

- **Easy to manage:** Each table can be independently manipulated without affecting others.
- **Security:** It is more secure consisting of multiple levels of security. Access of data shared can be limited.

- **Flexible:** Updating of data can be done at a single point without making amendments at multiple files. Databases can easily be extended to incorporate more records, thus providing greater scalability. Also, facilitates easy application of SQL queries.
- **Users:** RDBMS supports client-side architecture storing multiple users together.
- Facilitates storage and retrieval of large amount of data.
- **Easy Data Handling:**
    - Data fetching is faster because of relational architecture.
    - Data redundancy or duplicity is avoided due to keys, indexes, and normalization principles.
    - Data consistency is ensured because RDBMS is based on ACID properties for data transactions(Atomicity Consistency Isolation Durability).
- **Fault Tolerance:** Replication of databases provides simultaneous access and helps the system recover in case of disasters, such as power failures or sudden shutdowns.

Disadvantages of RDBMS

- **High Cost and Extensive Hardware and Software Support:** Huge costs and setups are required to make these systems functional.
- **Scalability:** In case of addition of more data, servers along with additional power and memory are required.
- **Complexity:** Voluminous data creates complexity in understanding of relations and may lower down the performance.
- **Structured Limits:** The fields or a column of a relational database system is enclosed within various limits, which may lead to loss of data.

Your Dreams Our Goal
POORNIMA UNIVERSITY
Member of Association of Indian Universities & Approved by UGC (Govt. of India) under 2(f) & 12(B)
DETAILED LECTURE NOTES

Lecture – 11		
Faculty of : FCE Program: BTECH	**Class/Section: IV**	**Date:**
Name of Faculty:-Mrs.Shakshi Ranawat	**Name of Course: RDBMS**	**Code: BCECCE4103**

Following are the various terminologies of RDBMS:

What is table/Relation?

Everything in a relational database is stored in the form of relations. The RDBMS database uses tables to store data. A table is a collection of related data entries and contains rows and columns to store data. Each table represents some real-world objects such as person, place, or event about which information is collected. The organized collection of data into a relational table is known as the logical view of the database.

Properties of a Relation:

- Each relation has a unique name by which it is identified in the database.
- Relation does not contain duplicate tuples.
- The tuples of a relation have no specific order.
- All attributes in a relation are atomic, i.e., each cell of a relation contains exactly one value.

A table is the simplest example of data stored in RDBMS.

Let's see the example of the student table.

ID	Name	AGE	COURSE
1	Ajeet	24	B.Tech

2	aryan	20	C.A
3	Mahesh	21	BTECH
4	Ratan	22	MCA
5	Vimal	26	BSC

What is a row or record?

A row of a table is also called a record or tuple. It contains the specific information of each entry in the table. It is a horizontal entity in the table. For example, The above table contains 5 records.

Properties of a row:

- No two tuples are identical to each other in all their entries.
- All tuples of the relation have the same format and the same number of entries.
- The order of the tuple is irrelevant. They are identified by their content, not by their position.

Let's see one record/row in the table.

ID	Name	AGE	COURSE
1	Ajeet	24	B.Tech

What is a column/attribute?

A column is a vertical entity in the table which contains all information associated with a specific field in a table. For example, "name" is a column in the above table which contains all information about a student's name.

Properties of an Attribute:

- Every attribute of a relation must have a name.
- Null values are permitted for the attributes.
- Default values can be specified for an attribute automatically inserted if no other value is specified for an attribute.
- Attributes that uniquely identify each tuple of a relation are the primary key.

Name
Ajeet
Aryan
Mahesh
Ratan
Vimal

What is data item/Cells?

The smallest unit of data in the table is the individual data item. It is stored at the intersection of tuples and attributes.

Properties of data items:

- Data items are atomic.
- The data items for an attribute should be drawn from the same domain.

In the below example, the data item in the student table consists of Ajeet, 24 and Btech, etc.

ID	Name	AGE	COURSE
1	Ajeet	24	B.Tech

Degree:

The total number of attributes that comprise a relation is known as the degree of the table.

For example, the student table has 4 attributes, and its degree is 4.

ID	Name	AGE	COURSE
1	Ajeet	24	B.Tech
2	aryan	20	C.A

3	Mahesh	21	BTECH
4	Ratan	22	MCA
5	Vimal	26	BSC

Cardinality:

The total number of tuples at any one time in a relation is known as the table's cardinality. The relation whose cardinality is 0 is called an empty table.

For example, the student table has 5 rows, and its cardinality is 5.

ID	Name	AGE	COURSE
1	Ajeet	24	B.Tech
2	aryan	20	C.A
3	Mahesh	21	BTECH
4	Ratan	22	MCA
5	Vimal	26	BSC

Domain:

The domain refers to the possible values each attribute can contain. It can be specified using standard data types such as integers, floating numbers, etc. **For example**, An attribute entitled Marital_Status may be limited to married or unmarried values.

NULL Values

The NULL value of the table specifies that the field has been left blank during record creation. It is different from the value filled with zero or a field that contains space.

Data Integrity

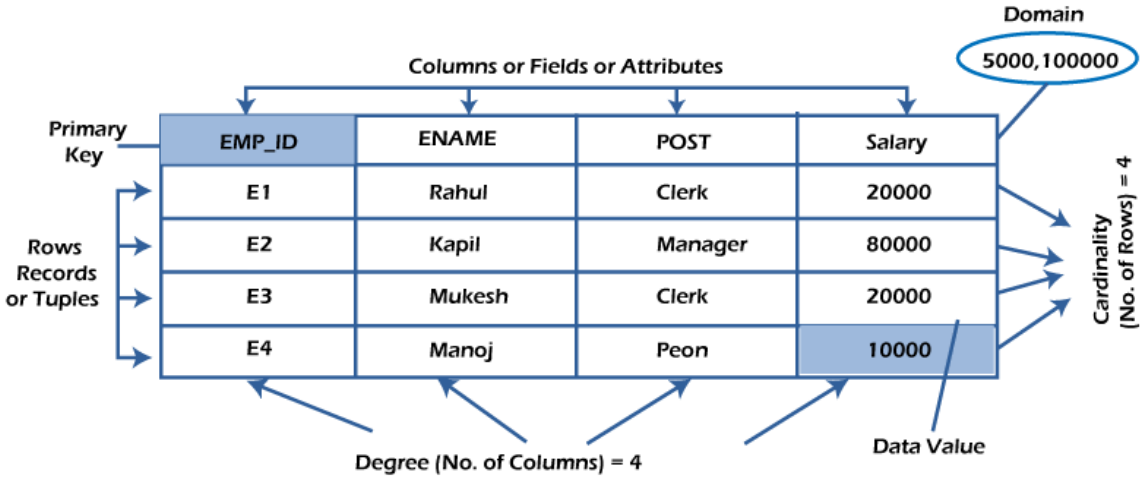
There are the following categories of data integrity exist with each RDBMS:

Entity integrity: It specifies that there should be no duplicate rows in a table.

Domain integrity: It enforces valid entries for a given column by restricting the type, the format, or the range of values.

Referential integrity specifies that rows cannot be deleted, which are used by other records.

User-defined integrity: It enforces some specific business rules defined by users. These rules are different from the entity, domain, or referential integrity.



- **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).

Your Dreams Our Goal

POORNIMA UNIVERSITY

Member of Association of Indian Universities & Approved by UGC (Govt. of India) under 2(f) & 12(B)

Detailed Lecture Notes

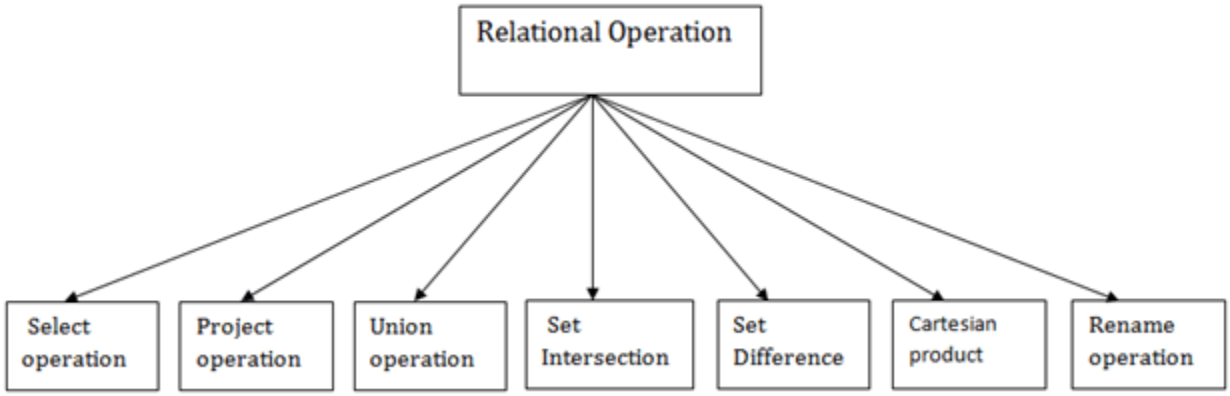
Your Dreams Our Goal
POORNIMA UNIVERSITY
Member of Association of Indian Universities & Approved by UGC (Govt. of India) under 2(f) & 12(B)
DETAILED LECTURE NOTES

Lecture – 12		
Faculty of : FCE Program: BTECH	**Class/Section: IV**	**Date:**
Name of Faculty:-Mrs.Shakshi Ranawat	**Name of Course: RDBMS**	**Code: BCECCE4103**

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.

Types of Relational operation



1. Select Operation:

- The select operation selects tuples that satisfy a given predicate.
- It is denoted by sigma (σ).

1 Notation: σ p(r)

Where:

σ is used for selection prediction

r is used for relation.
p is used as a propositional logic formula which may use connectors like: AND OR and NOT. These relational can use as relational operators like =, ≠, ≥, <, >, ≤.

For example: LOAN Relation

BRANCH_NAME	LOAN_NO	AMOUNT
Downtown	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

Input:

1 σ BRANCH_NAME="perryride" (LOAN)

Output:

BRANCH_NAME	LOAN_NO	AMOUNT
Perryride	L-15	1500
Perryride	L-16	1300

2. Project Operation:

- This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.
- It is denoted by ∏.

1 Notation: ∏ A1, A2, An (r)

Where

A1, **A2**, **A3** is used as an attribute name of relation **r**.

Example: CUSTOMER RELATION

NAME	STREET	CITY
Jones	Main	Harrison

Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

Input:

1 ∏ NAME, CITY (CUSTOMER)

Output:

NAME	CITY
Jones	Harrison
Smith	Rye
Hays	Harrison
Curry	Rye

Johnson	Brooklyn
Brooks	Brooklyn

3**. Union Operation:**

- Suppose there are two relation R and S. The union operation contains all the tuples that are either in R or S or both in R & S.
- It eliminates the duplicate tuples. It is denoted by ∪.

1 Notation: R ∪ S

A union operation must hold the following condition:

- R and S must have the attribute of the same number.
- Duplicate tuples are eliminated automatically.

Example:

DEPOSITOR RELATION

CUSTOMER_NAME	**ACCOUNT_NO**
Johnson	A-101
Smith	A-121

Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

BORROW RELATION

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93

Smith	L-11
Williams	L-17

Input:

1 ∏ CUSTOMER_NAME (BORROW) ∪ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson

Curry
Williams
Mayes

4. Set Intersection:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.
- It is denoted by intersection ∩.

1 Notation: R ∩ S

Example: Using the above DEPOSITOR table and BORROW table

Input:

1 ∏ CUSTOMER_NAME (BORROW) ∩ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Smith
Jones

5. Set Difference:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in R but not in S.
- It is denoted by intersection minus (-).

1 Notation: R - S

Example: Using the above DEPOSITOR table and BORROW table

Input:

1 ∏ CUSTOMER_NAME (BORROW) - ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Jackson
Hayes
Willians
Curry

6. Cartesian product

- The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.
- It is denoted by X.

1 Notation: E X D

Example:

EMPLOYEE

EMP_ID	EMP_NAME	EMP_DEPT
1	Smith	A
2	Harry	C
3	John	B

DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales

C	Legal

Input:

1 EMPLOYEE X DEPARTMENT

Output:

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	A	A	Marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Harry	C	A	Marketing
2	Harry	C	B	Sales
2	Harry	C	C	Legal
3	John	B	A	Marketing
3	John	B	B	Sales

3	John	B	C	Legal

7. Rename Operation:

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

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

1. ρ(STUDENT1, STUDENT)

Note: Apart from these common operations Relational algebra can be used in Join operations.

Join Operations:

A Join operation combines related tuples from different relations, if and only if a given join condition is satisfied. It is denoted by ⋈.

Example:

EMPLOYEE

EMP_CODE	EMP_NAME
101	Stephan
102	Jack
103	Harry

SALARY

EMP_CODE	SALARY
101	50000
102	30000
103	25000

1 Operation: (EMPLOYEE ⋈ SALARY)

Result:

EMP_CODE	EMP_NAME	SALARY
101	Stephan	50000
102	Jack	30000
103	Harry	25000

Your Dreams Our Goal

POORNIMA UNIVERSITY

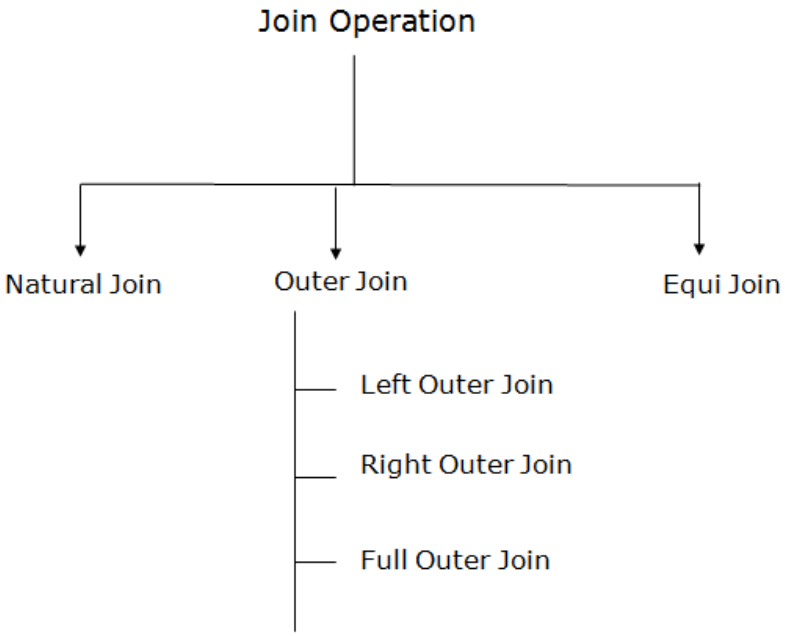
Member of Association of Indian Universities & Approved by UGC (Govt. of India) under 2(f) & 12(B)

Detailed Lecture Notes

Your Dreams Our Goal
POORNIMA UNIVERSITY
Member of Association of Indian Universities & Approved by UGC (Govt. of India) under 2(f) & 12(B)
DETAILED LECTURE NOTES

Lecture – 13		
Faculty of : FCE Program: BTECH	**Class/Section: IV**	**Date:**
Name of Faculty:-Mrs.Shakshi Ranawat	**Name of Course: RDBMS**	**Code: BCECCE4103**

Types of Join operations:



1. Natural Join:

- A natural join is the set of tuples of all combinations in R and S that are equal on their common attribute names.

- It is denoted by ⋈.

Example: Let's use the above EMPLOYEE table and SALARY table:

Input:

1 ∏EMP_NAME, SALARY (EMPLOYEE ⋈ SALARY)

Output:

EMP_NAME	SALARY
Stephan	50000
Jack	30000
Harry	25000

2. Outer Join:

The outer join operation is an extension of the join operation. It is used to deal with missing information.

Example:

EMPLOYEE

EMP_NAME	STREET	CITY

Ram	Civil line	Mumbai
Shyam	Park street	Kolkata
Ravi	M.G. Street	Delhi
Hari	Nehru nagar	Hyderabad

FACT_WORKERS

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Shyam	Wipro	20000
Kuber	HCL	30000
Hari	TCS	50000

Input:

1 (EMPLOYEE ⋈ FACT_WORKERS)

Output:

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru nagar	Hyderabad	TCS	50000

An outer join is basically of three types:

a. Left outer join
b. Right outer join
c. Full outer join

a. Left outer join:

- Left outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
- In the left outer join, tuples in R have no matching tuples in S.
- It is denoted by ⟕.

Example: Using the above EMPLOYEE table and FACT_WORKERS table

Input:

1 EMPLOYEE ⟕ FACT_WORKERS

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL

b. Right outer join:

- Right outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
- In right outer join, tuples in S have no matching tuples in R.
-
- It is denoted by ⋈.

Example: Using the above EMPLOYEE table and FACT_WORKERS Relation

Input:

1 EMPLOYEE ⋈ FACT_WORKERS

Output:

EMP_NAME	BRANCH	SALARY	STREET	CITY
Ram	Infosys	10000	Civil line	Mumbai
Shyam	Wipro	20000	Park street	Kolkata
Hari	TCS	50000	Nehru street	Hyderabad
Kuber	HCL	30000	NULL	NULL

c. Full outer join:

- Full outer join is like a left or right join except that it contains all rows from both tables.
- In full outer join, tuples in R that have no matching tuples in S and tuples in S that have no matching tuples in R in their common attribute name.
- It is denoted by ⟗.

Example: Using the above EMPLOYEE table and FACT_WORKERS table

Input:

1 EMPLOYEE ⟗ FACT_WORKERS

Output:

EMP_NAME	STREET	CITY	BRANCH	SALARY

Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL
Kuber	NULL	NULL	HCL	30000

3. Equi join:

It is also known as an inner join. It is the most common join. It is based on matched data as per the equality condition. The equi join uses the comparison operator(=).

Example:

CUSTOMER RELATION

CLASS_ID	NAME
1	John
2	Harry
3	Jackson

PRODUCT

PRODUCT_ID	CITY
1	Delhi
2	Mumbai
3	Noida

Input:

1 CUSTOMER ⋈ PRODUCT

Output:

CLASS_ID	NAME	PRODUCT_ID	CITY
1	John	1	Delhi
2	Harry	2	Mumbai
3	Harry	3	Noida

Your Dreams Our Goal
POORNIMA UNIVERSITY
Member of Association of Indian Universities & Approved by UGC (Govt. of India) under 2(f) & 12(B)
DETAILED LECTURE NOTES

Lecture – 14		
Faculty of : FCE Program: BTECH	**Class/Section: IV**	**Date:**
Name of Faculty:-Mrs.Shakshi Ranawat	**Name of Course: RDBMS**	**Code: BCECCE4103**

Normalization

A large database defined as a single relation may result in data duplication. This repetition of data may result in:

- Making relations very large.
- It isn't easy to maintain and update data as it would involve searching many records in relation.
- Wastage and poor utilization of disk space and resources.
- The likelihood of errors and inconsistencies increases.

So to handle these problems, we should analyze and decompose the relations with redundant data into smaller, simpler, and well-structured relations that are satisfy desirable properties. Normalization is a process of decomposing the relations into relations with fewer attributes.

What is 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 also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

Why do we need Normalization?

The main reason for normalizing the relations is removing these anomalies. Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows. Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

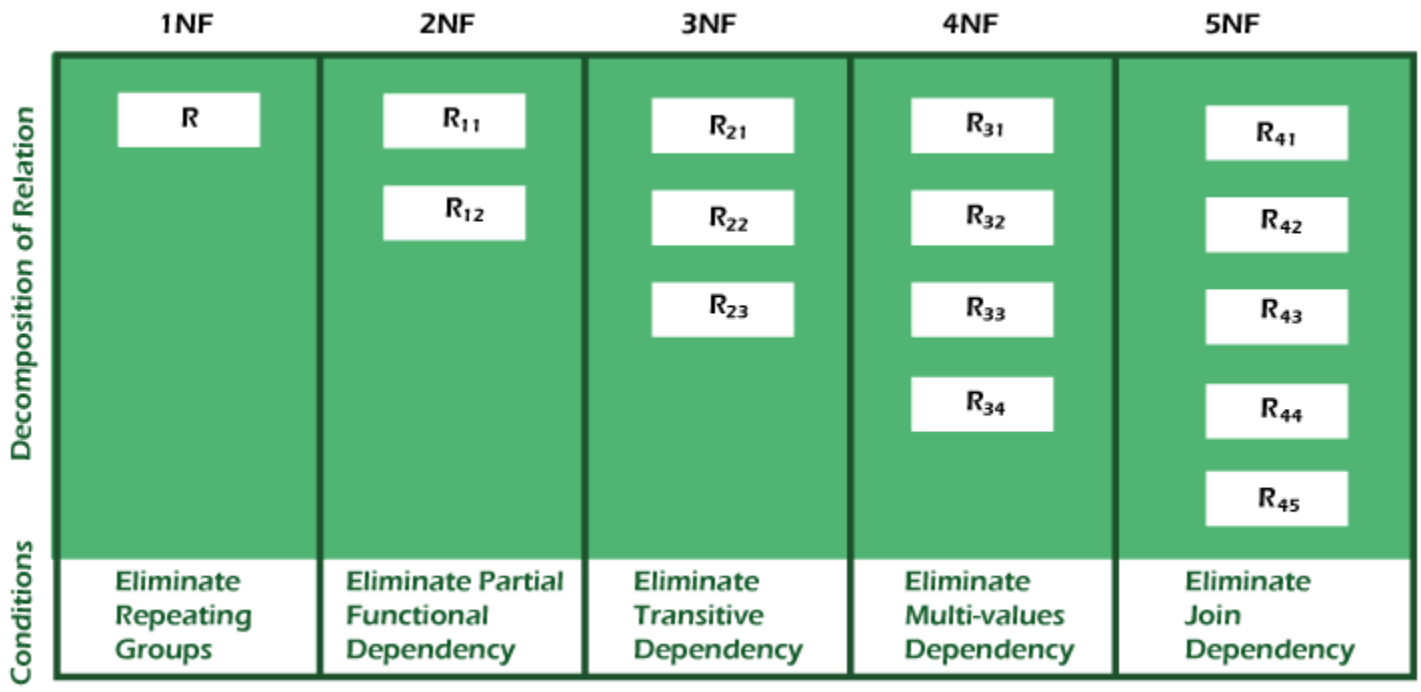
Data modification anomalies can be categorized into three types:

- **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
- **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
- **Updatation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

Types of Normal Forms:

Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints.

Following are the various types of Normal forms:



Normal	Description

Form	
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
BCNF	A stronger definition of 3NF is known as Boyce Codd's normal form.
4NF	A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency.
5NF	A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless.

Advantages of Normalization

- Normalization helps to minimize data redundancy.
- Greater overall database organization.
- Data consistency within the database.
- Much more flexible database design.
- Enforces the concept of relational integrity.

Disadvantages of Normalization

- You cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.

- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to serious problems.

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 hold only single-valued attribute.
- First normal form 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, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

The decomposition of the EMPLOYEE table into 1NF has been shown below:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab

12	Sam	8589830302	Punjab

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, non-prime attribute TEACHER_AGE 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

Third Normal Form (3NF)

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

A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency X → 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:

EMPLOYEE_DETAIL table:

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
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

Super key in the table above:

1. {EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID, EMP_NAME, EMP_ZIP}....so on

   Candidate key: {EMP_ID}

   Non-prime attributes: In the given table, all attributes except EMP_ID are non-prime.

   Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on super key(EMP_ID). It violates the rule of third normal form.

   That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

   EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389

666	John	462007

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

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 → Y, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Example: Let's assume there is a company where employees 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	D283	232
364	UK	Developing	D283	549

In the above table Functional dependencies are as follows:

1. EMP_ID → EMP_COUNTRY
2. 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 into BCNF, we decompose it into three tables:

EMP_COUNTRY table:

EMP_ID	EMP_COUNTRY
264	India
264	India

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT

D394	283
D394	300
D283	232
D283	549

Functional dependencies:

1. EMP_ID → EMP_COUNTRY
2. EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate keys:

For the first table: EMP_ID
For the second table: EMP_DEPT
For the third table: {EMP_ID, EMP_DEPT}

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

Fourth normal form (4NF)

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

Example

STUDENT

STU_ID	COURSE	HOBBY
21	Computer	Dancing

21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.

So to make the above table into 4NF, we can decompose it into two tables:

STUDENT_COURSE

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

STU_ID	HOBBY

21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

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 into 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 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.

Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

P1

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

P3

Your Dreams Our Goal

POORNIMA UNIVERSITY

Member of Association of Indian Universities & Approved by UGC (Govt. of India) under 2(f) & 12(B)

DETAILED LECTURE NOTES

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen

Lecture – 15		
Faculty of : FCE Program: BTECH	**Class/Section: IV**	**Date:**
Name of Faculty:-Mrs.Shakshi Ranawat	**Name of Course: RDBMS**	**Code: BCECCE4103**

Inference Rule (IR):

- The Armstrong's axioms are the basic inference rule.
- Armstrong's axioms are used to conclude functional dependencies on a relational database.
- The inference rule is a type of assertion. It can apply to
- .a set of FD(functional dependency) to derive other FD.
- Using the inference rule, we can derive additional functional dependency from the initial set.

The Functional dependency has 6 types of inference rule:

1. Reflexive Rule (IR_1)

In the reflexive rule, if Y is a subset of X, then X determines Y.

1 If $X \supseteq Y$ then $X \rightarrow Y$

Example:

1 X = {a, b, c, d, e}
2 Y = {a, b, c}

2. Augmentation Rule (IR_2)

The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z.

1 If $X \rightarrow Y$ then $XZ \rightarrow YZ$

Example:

1 For R(ABCD), **if** A → B then AC → BC

3. Transitive Rule (IR_3)

In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.

1 If X → Y and Y → Z then X → Z

4. Union Rule (IR_4)

Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.

1 If X → Y and X → Z then X → YZ

Proof:

1. X → Y (given)
2. X → Z (given)
3. X → XY (using IR_2 on 1 by augmentation with X. Where XX = X)
4. XY → YZ (using IR_2 on 2 by augmentation with Y)
5. X → YZ (using IR_3 on 3 and 4)

5. Decomposition Rule (IR_5)

Decomposition rule is also known as project rule. It is the reverse of union rule.

This Rule says, if X determines Y and Z, then X determines Y and X determines Z separately.

1 If X → YZ then X → Y and X → Z

Proof:

1. X → YZ (given)
2. YZ → Y (using IR_1 Rule)
3. X → Y (using IR_3 on 1 and 2)

6. Pseudo transitive Rule (IR_6)

In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.

1 If X → Y and YZ → W then XZ → W

Proof:

1. X → Y (given)
2. WY → Z (given)
3. WX → WY (using IR_2 on 1 by augmenting with W)
4. WX → Z (using IR_3 on 3 and 2)