2. Create a word documentation of Introduction to DBMS,RDMS

- Include all points that we have discussed so far.

- Include additional important points from your research related to DBMS

- Include Advantages and disadvantages of DBMS and RDMS with examples

- Include different other types of normalization techniques.

**DATABASE**

* Database is a collection of related data and data is a collection of facts and figures that can be processed to produce information.
* Mostly data represents recordable facts.
* Data aids in producing information, which is based on facts.

**What is RDBMS?**

RDBMS stands for Relational Database Management System.

RDBMS is a program used to maintain a relational database.

RDBMS is the basis for all modern database systems such as MySQL, Microsoft SQL Server, Oracle, and Microsoft Access.

**Database Management System (DBMS)**

**DBMS** is a software that is used to define, create, and maintain a database and provides controlled access to the data.

**Why is DBMS Required?**

Database management system, as the name suggests, is a management system that is used to manage the entire flow of data, i.e, the insertion of data or the retrieval of data, how the data is inserted into the database, or how fast the data should be retrieved, so DBMS takes care of all these features, as it maintains the uniformity of the database as well does the faster insertions as well as retrievals.

Database management system is software which is used for creating and managing different databases

**With the help of DBMS, we take care of following things**

* Data Security
* Data Backup
* Manages huge amount of data
* Data export and import
* Serving multiple concurrent database requests
* Gives us a way to manage the data using programming languages

**Why is RDBMS Required?**

RDBMS on the other hand is a type of DBMS, as the name suggests it deals with relations as well as various key constraints. So here we have tables which are called schema and we have rows which are called tuples. It also aids in the reduction of data redundancy and the preservation of database integrity.

Relational Database Management System is an **advanced** version of a DBMS.

* The most popular data model in DBMS is the Relational Model. It is more scientific model than others
* Data is stored in tables called relations.
* Relations can be normalized.
* In normalized relations, values saved are atomic values.
* Each row in a relation contains a unique value.
* Database is a collection of related data and data is a collection of facts and figures that can be processed to produce information.
* Mostly data represents recordable facts.
* Data aids in producing information, which is based on facts.
* A Relational Database management System(RDBMS) is a database management system based on relational data model introduced by E.F Codd.
* Relational data model is the primary data model, which is used widely around the world for data storage and processing.
* This model is simple, and it has all the properties and capabilities required to process data with storage efficiency.
* RDBMS is the basis for SQL and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
* Relational database is a collection of organized set of tables from which data can be accessed easily.
* An RDBMS can be defined as a DBMS where all the data is organized strictly as tables of data values .
* Each table in a database has a unique table name.
* A record, also called a row of data, is each individual entry that exists in a table
* The row (or record) in the table is called a tuple.
* The column (or field) is called an attribute.
* Every column in a table must have a unique name.
* The number of tuples is called the cardinality of the table.
* The number of attributes is called the degree of the table.
* A single entry in a table is called a Record or Row.
* A Record in a table represents set of related data SQL Constraints
* Constraints are the rules enforced on data columns on table.
* These are used to limit the type of data that can go into a table.

This ensures the accuracy and reliability of the data in the database. Some of the mostly used constraints are;

* NOT NULL : Ensures that a column cannot have NULL value.
* UNIQUE : Ensures that all values in a column are different. Ø PRIMARY Key: Uniquely identified each rows/records in a database table.
* FOREIGN Key: Uniquely identified a rows/records in any another database table.

**Difference Between DBMS and RDBMS**

| **DBMS** | **RDBMS** |
| --- | --- |
| [DBMS](https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/) stores data as file. | [RDBMS](https://www.geeksforgeeks.org/rdbms-architecture/) stores data in tabular form. |
| Data elements need to access individually. | Multiple data elements can be accessed at the same time. |
| No relationship between data. | Data is stored in the form of tables which are related to each other. |
| Normalization is not present. | Normalization is present. |
| DBMS does not support distributed database. | RDBMS supports distributed database. |
| It stores data in either a navigational or hierarchical form. | It uses a tabular structure where the headers are the column names, and the rows contain corresponding values. |
| It deals with small quantity of data. | It deals with large amount of data. |
| Data redundancy is common in this model. | Keys and indexes do not allow Data redundancy. |
| It is used for small organization and deal with small data. | It is used to handle large amount of data. |
| Not all Codd rules are satisfied. | All 12 Codd rules are satisfied. |
| Security is less | More security measures provided. |
| It supports single user. | It supports multiple users. |
| Data fetching is slower for the large amount of data. | Data fetching is fast because of relational approach. |
| The data in a DBMS is subject to low security levels with regards to data manipulation. | There exists multiple levels of data security in a RDBMS. |
| Low software and hardware necessities. | Higher software and hardware |

**Advantages of RDBMS**

Relational Database Management Systems (RDBMS) offer numerous advantages, making them widely used in various applications and industries. Here are some of the key advantages of RDBMS:

1. Data Structure

They organize data into tables with rows and columns, providing a structured and consistent way to store and manage data. This structure facilitates easy access, retrieval, and manipulation of data.

2. Data Integrity

They enforce data integrity through constraints, such as primary keys, foreign keys, and unique constraints. These constraints ensure that the data remains accurate and consistent, preventing the introduction of duplicate or invalid information.

3. Data Relationships

They enable the establishment of relationships between tables using primary and foreign keys. This feature allows data from different tables to be linked together, making it easier to query and analyze related information.

4. Querying and Reporting

They provide powerful query languages (e.g., SQL) that allow users to retrieve specific data from the database quickly. This simplicity and flexibility make it easier to generate reports and gain insights from the data.

5. Data Security

They offer robust security mechanisms to protect sensitive data. Access control features allow administrators to define user permissions and restrict unauthorized access to data.

6. Scalability

They can handle large amounts of data and scale up to accommodate the growth of the dataset and user demand. Many RDBMS platforms offer high-availability options and clustering to ensure system availability even during peak usage.

7. ACID Transactions

They support [ACID](https://trainings.internshala.com/blog/acid-properties-in-dbms/) (Atomicity, Consistency, Isolation, Durability) transactions, which guarantee that database operations are reliable and that data remains in a consistent state even in the event of failures.

Also, explore the key [difference between DBMS and RDBMS](https://trainings.internshala.com/blog/difference-between-dbms-rdbms/).

8. Data Backup and Recovery

They provide tools and mechanisms to perform regular backups of the database, enabling data recovery in case of hardware failures, accidental deletions, or other unforeseen issues.

9. Data Indexing

They use indexing techniques to optimize data retrieval. Indexes speed up query execution by reducing the need for full-table scans, especially for large datasets.

10. Data Normalization

They encourage data normalization, which helps eliminate data redundancy and inconsistencies. Normalized data ensures efficient storage and minimizes update anomalies.

11. Industry Standard

RDBMS, such as [MySQL](https://trainings.internshala.com/blog/what-is-mysql/), Oracle, Microsoft SQL Server, and PostgreSQL, have been around for decades and have established themselves as reliable, widely used, and well-supported solutions.

12. Data Consistency

They enforce referential integrity, ensuring that relationships between data remain consistent. This prevents data inconsistencies and maintains the accuracy and reliability of the information stored in the database.

**13. Data Independence**

They offer data abstraction, separating the logical structure of the database from its physical implementation. This data independence allows developers to modify the database schema without affecting the applications built on top of it, promoting flexibility and reducing maintenance efforts.

14. Data Accessibility

They provide concurrent access to multiple users and applications. Users can read and write data simultaneously without interfering with each other’s operations, promoting efficient collaboration and multi-user support.

15. Data Backup and Recovery

They typically offer various backup and recovery options, including full, incremental, and differential backups. These features ensure data protection and the ability to restore the database to a specific point in time.

**Disadvantages of RDBMS**

1. Scalability Limitations

RDBMS may face challenges when dealing with extremely large datasets and high transaction volumes. As the data size grows, the performance of the RDBMS can degrade, requiring careful database design and optimization to maintain efficiency.

2. Complex Design

Designing a relational database schema can be complex, especially for large and intricate applications. Ensuring proper normalization and establishing appropriate relationships between tables can be time-consuming and require significant expertise.

3. Fixed Schema

They have a fixed schema, meaning the structure of the database is defined beforehand. Adding new columns or altering the schema often requires modifying existing applications, which can be cumbersome and may lead to downtime during updates.

4. Performance Bottlenecks

Certain operations, such as complex joins, can lead to performance bottlenecks in RDBMS, especially when dealing with large datasets. Proper indexing and query optimization are essential to mitigate these issues.

5. High Overhead

They typically have more overhead compared to some NoSQL databases. The need for data normalization, transactions, and referential integrity enforcement can result in increased storage requirements and slower performance.

6. Cost

Some commercial RDBMS solutions can be expensive, especially when considering licensing, maintenance, and hardware requirements. While open-source options like [MySQL](https://trainings.internshala.com/blog/what-is-mysql/) and [PostgreSQL](https://trainings.internshala.com/blog/what-is-postgresql/) exist, implementing and managing RDBMS still incurs costs.

7. Replication Complexity

While RDBMS supports replication, setting up and managing replication can be complex, particularly in distributed and multi-data center environments.

8. Single Point of Failure

In traditional RDBMS setups, the database server can become a single point of failure. Ensuring high availability often requires implementing clustering or failover mechanisms, which can add complexity to the system.

9. Data Modeling Challenges

Representing certain types of data, such as hierarchical or unstructured data, can be challenging in a relational database. NoSQL databases may be better suited for handling these specific data types.

10. Data Type Limitations

RDBMS have predefined data types, and accommodating certain data formats or unstructured data can be difficult without resorting to workarounds.

11. Concurrency Issues

In heavily concurrent systems, managing locks and ensuring data consistency can be complex and may lead to potential contention and performance bottlenecks.

12. Learning Curve

They require users to be familiar with complex [SQL](https://www.oracle.com/database/sqldeveloper/technologies/download/) queries and relational algebra. Mastering these skills can take time, especially for individuals new to database management.

13. Inflexibility with Changing Requirements

When the application’s requirements change frequently, altering the relational schema can be time-consuming and disruptive.

14. Complex Joins and Performance

While relational databases are excellent at handling structured data and enforcing data integrity, complex joins involving multiple tables can become a performance challenge. As the number of tables and the complexity of relationships increase, query execution times may also increase significantly.

Advantages of DBMS

The use of a database management system, or DBMS, to store and manage data has several advantages. These are DBMS's advantages:

**Improves the effectiveness of data exchange**

With DBMS, data can be exchanged between users more effectively, and access to the data can be restricted so that only authorized users are permitted to view it, as opposed to earlier systems when everyone with access to the system could access the data. We can more easily manage the data in a DBMS.

**Heightens Data Protection**

Data is now one of the most precious resources available in the modern world. Additionally, the need for data protection becomes even more critical. A large amount of people having access to the database raises the likelihood that the data may be compromised. A simple security layout can be provided by the database management system. Only users with such permissions will be able to view or alter the data, according to limits placed on the information's access by the database administrator. Although it does not guarantee total security, it does offer a solid security design.

**Safeguarding Data Integrity**

It is essential to offer specific capabilities, such as executing numerous transactions and allowing continuous access to the data, when giving many users database access. Maintaining the accuracy of the information is essential to prevent data loss when numerous users attempt to alter the same piece of data at the same time. Data redundancy is reduced in the database by the normalized format in which the data is kept. Additionally, it lessens any discrepancies in the data. Inside a database, the entire set of data is kept in a single file, as opposed to a file system in which it is spread across numerous directories, files, and folders.

**Enhance the Process of Decision-Making**

It is considerably simpler to study the data because it is presented in a more organized format with rows and columns by the DBMS. We can reach certain conclusions by doing straightforward database queries. Constraints that must be followed when storing data in DBMS improve data quality, which in turn improves decision-making. The productivity and utility of the data improve dramatically as a result.

**Recovery and Back-up**

Data is the most precious resource for the entity, as was described before; therefore, data preservation is just as critical as data protection. By performing regular backups using a DBMS, a user can store the most recent data on the drive or the cloud. The user can utilize the restore to retrieve the information from the drive or even the cloud if it is deleted from the system.

Disadvantages of DBMS

Although DMBS provides a lot of benefits, it also has a lot of drawbacks. DBMS has the following drawbacks:

**Specifications for Hardware and Software**

A system with a high configuration is needed to operate the DBMS effectively. We will unavoidably need hardware that performs well to get this height. As all of this technology and the license for this program are relatively pricey, it raises the cost of development. On your local system, they also take up comparatively more room. Also necessary is the upkeep of these systems.

**Management scope and complexity**

Due to the large range of functions, it offers, the database project's scalability is increased. To create a user interface, it supports many GUIs. It may also be used in conjunction with other potent software. But the complexities of the system as a whole are increased by this entire situation. The process is highly complicated as a result of all these implementations. We need to know other SQL languages to maintain the data and operate the database.

**Huge Dimensions**

For database management software to work correctly, a lot of disc space is needed. It needs extra software, and that software needs storage space. Gigabytes of space may be needed for the whole DBMS configuration.

**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 Form** | **Description** |
| [1NF](https://www.javatpoint.com/dbms-first-normal-form) | A relation is in 1NF if it contains an atomic value. |
| [2NF](https://www.javatpoint.com/dbms-second-normal-form) | 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](https://www.javatpoint.com/dbms-third-normal-form) | 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](https://www.javatpoint.com/dbms-forth-normal-form) | A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency. |
| [5NF](https://www.javatpoint.com/dbms-fifth-normal-form) | 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**

|  |  |
| --- | --- |
| **SEMSTER** | **LECTURER** |
| Semester 1 | Anshika |
| Semester 1 | John |
| Semester 1 | John |
| Semester 2 | Akash |
| Semester 1 | Praveen |

THANK YOU