

# Managing and Programming Databases

**MCDA5540**

**PROJECT – 1**

DMBS Project Design

# SUBMITTED BY:

**DIVEN KUMAR SAMBHWANI(A00425905)**

**JASLEEN KOUR(A00425843)**

**RAVNEET SINGH OBEROI (A00426623)**

**SONAM VADSARIA(A00431604)**

# Masters in Computing and Data Analytics

**SUBMITTED TO: TRISHLA SHAH**

1. Database Design :

Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data to the database model. Database design involves classifying data and identifying interrelationships.

* 1. Good Database design :

Certain principles guide the database design process. The first principle is that duplicate information (also called redundant data) is bad, because it wastes space and increases the likelihood of errors and inconsistencies. The second principle is that the correctness and completeness of information is important. If your database contains incorrect information, any reports that pull information from the database will also contain incorrect information. As a result, any decisions you make that are based on those reports will then be misinformed.

A good database design is, therefore, one that:

* Divides your information into subject-based tables to reduce redundant data.
* Provides Access with the information it requires to join the information in the tables together as needed.
* Helps support and ensure the accuracy and integrity of your information.
* Accommodates your data processing and reporting needs.

1. ENTITY-RELATIONSHIP DIAGRAM(ERD)

One of the best design approaches is Entity Relationship Method. This design approach is widely followed in designing projects normally known as “Entity Relationship Diagram (ERD)”.

ERD helps in capturing the business rules governing the data relationships of the system and is a conventional aid for communicating with the end users in the conceptual design phase. ERD consists of:

* **Entity** – It is the term use to describe any object, place, person, and concept, activity that the enterprise recognizes in the area under investigation and wishes to collect and store data. It is diagrammatically represented as boxes.
* **Attribute** – They are the data elements that are used to describe the properties that distinguish the entities.
* **Relationship** – It is an association or connection between two or more entities. They are diagrammatically represented as arrows.

A unary relationship is a relationship between instances of the same entity.

A binary relationship is a relationship between two entities.

A n-ary relationship is a relationship among n entities. It is defined only when the relationship does have a meaning without the participation of all the n entities.

* **Degree of relationship** – an important aspect of relationship between two or more entities is the degree of relationship. The different relationships recognized among various data stores in the database are:
* **One-to-one (1:1)**

It is an association between two entities. For example, each student can have only one roll no.

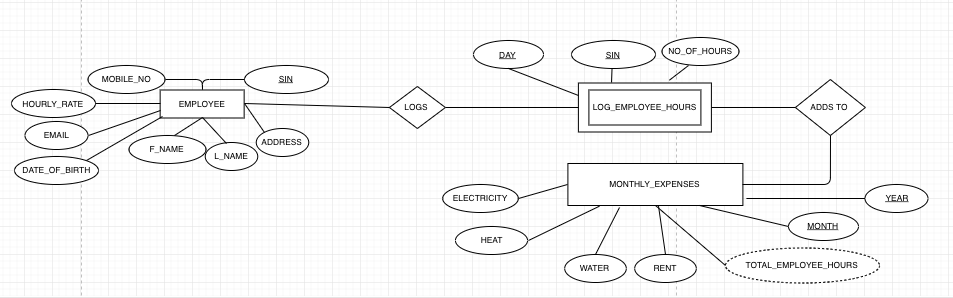
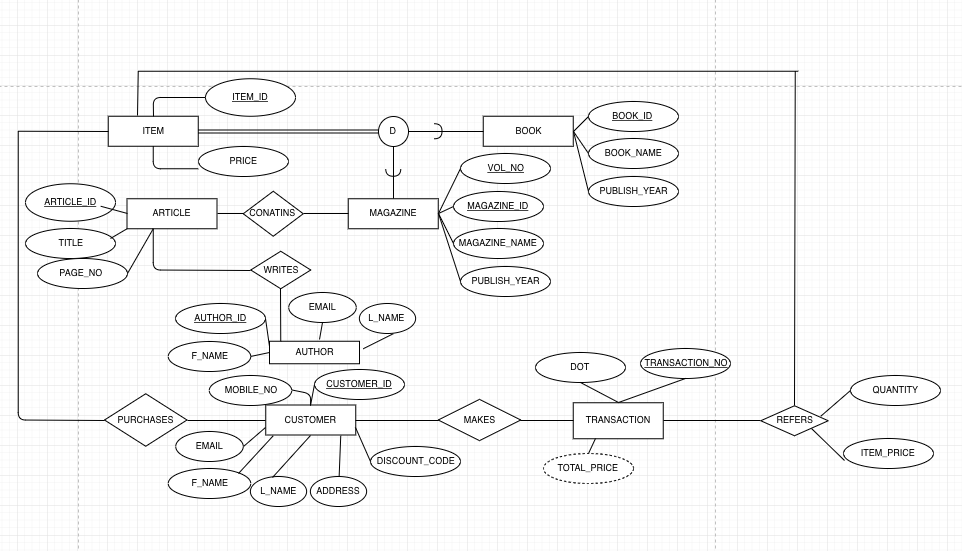
* **One-to-many (1:m)**

It describes entities that may have one or more entities related to it. For example, a father may have one or many children.

* **Many-to-many (m:m)**

It describes entities that may have relationships in both directions. This relationship can be explained by considering items sold by vendors. A vendor can sell many items and many vendors can sell each item.

**ERR representation of the project is given below.**



1. **Normalization :** Normalization is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data. It divides larger tables to smaller tables and links them using relationships. To achieve a refined design, you must **normalize**your tables.
   1. **Normalization Rules:**

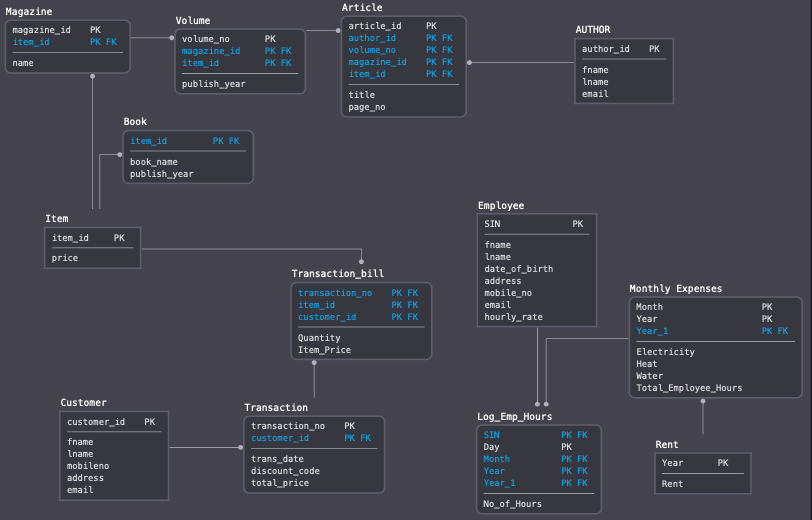
**1NF -First Normal Form:**

•Rule-1: **Single Valued Attribute**. Each column of your table should be single valued which means they should not contain multiple values.

•Rule -2 : **Attribute domain should not change**. In each column the values stored must be of the same kind or type.

•Rule-3: **Unique name for Attributes/Columns**. This rule expects that each column in a table should have a unique name

•Rule-4: **Order doesn’t matter.**



**2NF -Second Normal Form :**

For a table to be in the Second Normal Form, it must satisfy two conditions:

1) The table should be in the First Normal Form.

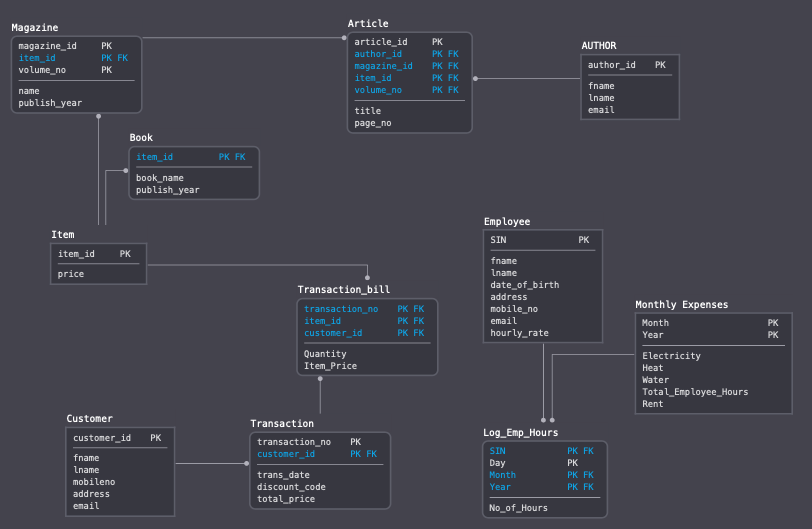
2) There should be no Partial Dependency

**3NF -Third Normal Form :**

•Eliminate fields that do not depend on the key.

•Values in a record that are not part of that record's key do not belong in the table.

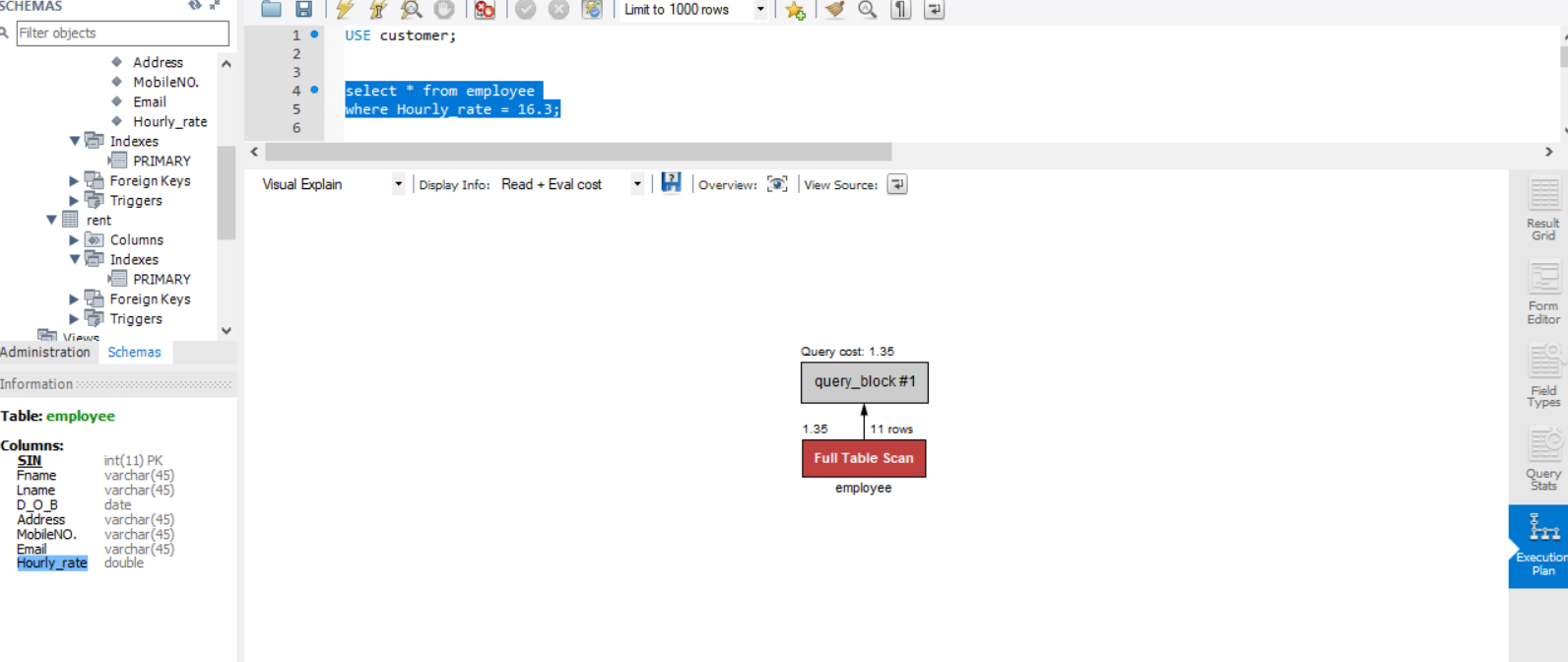
•In general, any time the contents of a group of fields may apply to more than a single record in the table, consider placing those fields in a separate table.



* **Item**: No Partial or transitive dependency as price is unique for each id
* **Magazine**: All the fields are dependent on id and volume no and there’s no partial dependency nor transitive dependency as it depends on id + volume no both and not the other way
* **Book**: No Partial or transitive dependency as all the rows are unique for each id
* **Article** : All the fields are dependent on article\_id ,volume no and Magazine id. There’s no partial dependency nor transitive dependency as each row depends on article id + volume no + magazine\_id and not the other way
* **Author**: No Partial or transitive dependency as each row is unique for each author\_ id
* **Customer**: No Partial or transitive dependency as each row is unique for each customer\_id
* **Transaction:** All fields are dependent on Transaction\_no, and there’s no partial or transitive dependency
* **Transaction\_bill:** All fields are dependent on Transaction\_no+Item\_id, . There’s no partial dependency nor transitive dependency as each row depends on Transaction\_no+Item\_id and not the other way
* **Employee**: No Partial or transitive dependency as each row is unique for each SIN
* **Monthly Expenses:** All fields are dependent on Month+Year . There’s no partial dependency nor transitive dependency as each row depends on Month+Year and not the other way
* **Log\_Emp\_Hours**: No Partial or transitive dependency as No\_of\_Hours is unique for SIN+DAY+MONTH+YEAR
  1. **Query Optimization :**

The goal of **query optimization** is to reduce the system resources required to fulfill a **query**, and ultimately provide the user with the correct result set faster. First, it provides the user with faster results, which makes the application seem faster to the user.

Here we have taken table ‘employee’ to show the query execution plan.



**Figure : Execution Plan without Indexing.**

According to the data in the table ‘employee’ there is only one employee with Hourly\_rate = 16.3 which is unique therefore result comes out to be only one row. But we can see in the execution plan that shows that Full Table scan has been done. Indexing the ‘employee’ table on ‘Hourly\_rate’. After indexing we can clearly see in the figure below that there is reduction in Query cost.

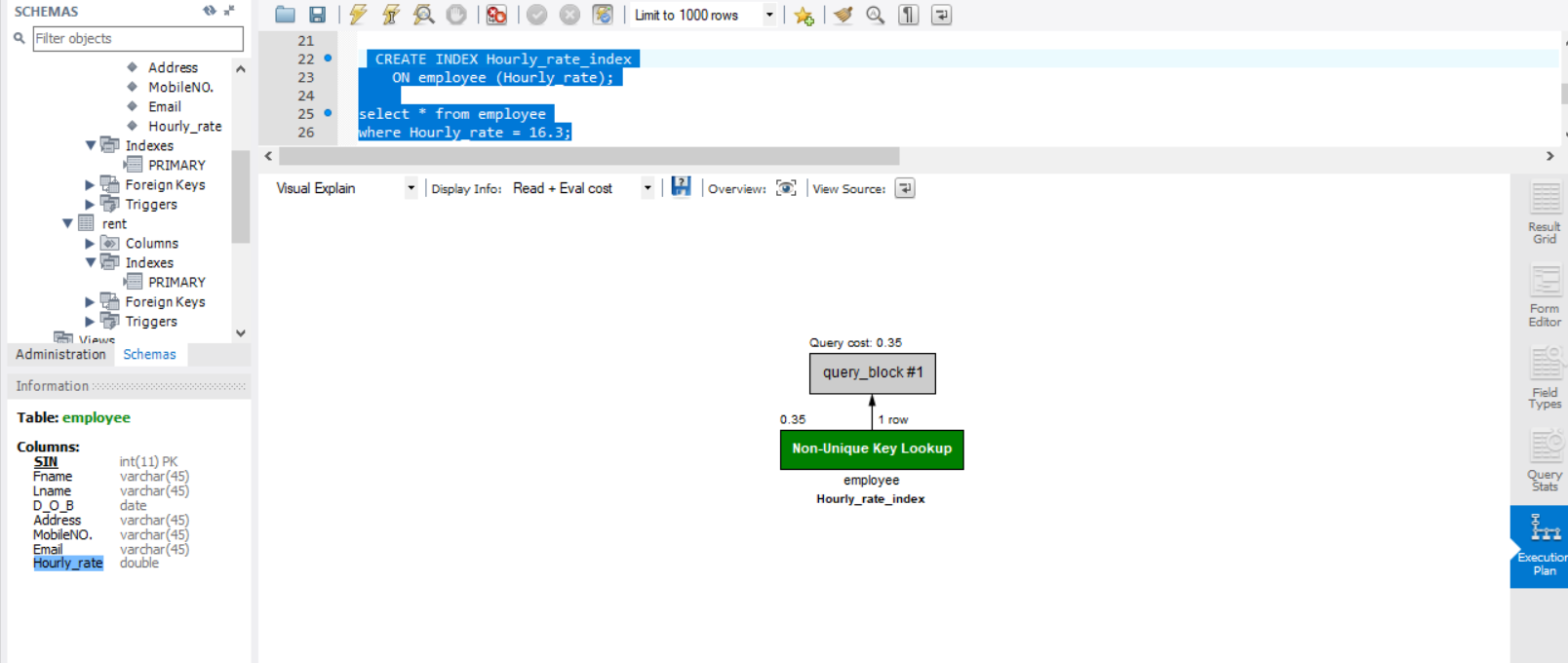
**Query :**

CREATE INDEX Hourly\_rate\_index

ON employee (Hourly\_rate);

select \* from employee

where Hourly\_rate = 16.3;



**Figure : Query Optimization**

4.5. **References :**

* <https://support.office.com/en-us/article/database-design-basics-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5>
* <https://medium.com/@kimtnguyen/relational-database-schema-design-overview-70e447ff66f9>
* <https://smu.brightspace.com/d2l/le/content/43109/viewContent/434128/View>
* <https://smu.brightspace.com/d2l/le/content/43109/viewContent/438528/View>
* <https://www.geeksforgeeks.org/query-optimization/>