**Document the steps to create logical design**

**Logical Design**

Logical design is the second stage in the database design process. The logical design goal is to

design an enterprise-wide database based on a specific data model but independent of physicallevel details. Logical design requires that all objects in the conceptual model be mapped to the

specific constructs used by the selected database model. For example, the logical design for a

relational DBMS includes the specifications for the relations (tables), relationships, and constraints

(i.e., domain definitions, data validations, and security views).

**The logical design is generally performed in four steps, which are as follows**.

1.Map conceptual model to logical model components

2.Validate logical model using normalization

3.Validate logical model integrity constraints

4. Validate logical model against user requirements

**1. Map the Conceptual Model to the Logical Model:**

The first step in creating the logical design is to map the conceptual model to the chosen database

constructs. Logical design generally involves translating the ER model into a set of relations

(tables), columns, and constraints definitions. The process of translating the conceptual model into

a set of relations is depicted as follows.

1. Map strong entities

2. Map super type

3. Map weak entity

4. Map binary relationships

5. Map higher degree relationships

**2. Validate the Logical Model Using Normalization:**

The logical design should contain only properly normalized tables. The process of mapping the

conceptual model to the logical model may unveil some new attributes or the discovery of new

multivalued or composite attributes. Therefore, it’s very likely that new attributes may be added to

tables or entire new tables added to the logical model. For each identified table (old and new), you

must ensure that all attributes are fully dependent on the identified primary key and that the tables are

in at least third normal form (3NF

**3. Validate Logical Model Integrity Constraints**:

The translation of the conceptual model into a logical model also requires the definition of the

attribute domains and appropriate constraints. For example, the domain definitions for the

CLASS\_CODE, CLASS\_DAYS, and CLASS\_TIME attributes of the CLASS entity are written this

way:

CLASS\_CODE is a valid class code.

**4. Validate the Logical Model against User Requirements:**

The logical design translates the software-independent conceptual model into a softwaredependent model. The final step in the logical design process is to validate all logical model

definitions against all end-user data, transaction, and security requirements. The stage is now set

to define the physical requirements that allow the system to function within the selected

DBMS/hardware environment.

**ACTIVITY -3**

**Discuss the document codd’s 12 rule**

**Codd’s Rule**

• Rule 1: Information Rule

The data stored in a database, may it be user data or metadata, must be a value of some table cell.

Everything in a database must be stored in a table format.

• Rule 2: Guaranteed Access Rule

Every single data element (value) is guaranteed to be accessible logically with a combination of

table-name primary-key (row value), and attribute-name (column value). No other means, such as

pointer can be used to access data.

• Rule 3: Systematic Treatment of NULL Values

The NULL values in a database must be given a systematic and uniform treatment. This is a very

Important rule because a NULL can be interpreted as one the following − data is missing, data is

not known, or data is not applicable.

• Rule 4: Active Online Catalog

The structure description of the entire database must be stored in an online catalog, known as data

dictionary, which can be accessed by authorized users. Users can use the same query language to

access the catalog which they use to access the database itself.

• Rule 5: Comprehensive Data Sub-Language Rule

A database can only be accessed using a language having linear syntax that supports data

definition, data manipulation, and transaction management operations. This language can be used

directly or by means of some application. If the database allows access to data without any help of

this language, then it is considered as a violation.

• Rule 6: View Updating Rule

All the views of a database, which can theoretically be updated, must also be updatable by the

system.

• Rule 7: High-Level Insert, Update, and Delete Rule

A database must support high-level insertion, updation, and deletion. This must not be limited to

a single row, that is, it must also support union, intersection and minus operations to yield sets of

data records.

• Rule 8: Physical Data Independence

The data stored in a database must be independent of the applications that access the database. Any

change in the physical structure of a database must not have any impact on how the data is being

accessed by external applications.

• Rule 9: Logical Data Independence

The logical data in a database must be independent of its user’s view (application). Any change in

logical data must not affect the applications using it. For example, if two tables are merged or one

is split into two different tables, there should be no impact or change on the user application. This

is one of the most difficult rule to apply.

• Rule 10: Integrity Independence

A database must be independent of the application that uses it. All its integrity constraints can be

independently modified without the need of any change in the application. This rule makes a

database independent of the front-end application and its interface.

• Rule 11: Distribution Independence

The end-user must not be able to see that the data is distributed over various locations. Users should

always get the impression that the data is located at one site only. This rule has been regarded as

the foundation of distributed database systems.

• Rule 12: Non-Subversion Rule

If a system has an interface that provides access to low-level records, then the interface must not

be able to subvert the system and bypass security and integrity constraints.