NORMALIZATION

Good database design must be matched to good table structures.

Normalization is a process for evaluating and correcting table structures to minimize data redundancies, thereby reducing the likelihood of data anomalies. Normalization is a database design technique, which begins by examining the relationships (called functional dependencies) between attributes.

The normalization process involves assigning attributes to tables based on the concept of determination.

Normalization works through a series of stages called normal forms.

The first three stages are described as

- First normal form (1NF),
- Second normal form (2NF),
- Third normal form (3NF).

THE NEED FOR NORMALIZATION

- After the initial design is complete, the designer can use normalization to analyze the relationships that exist among the attributes within each entity, to determine if the structure can be improved through normalization.
- Alternatively, database designers are often asked to modify existing data structures that can be in the form of flat files, spreadsheets, or older database structures.
- Again, through an analysis of the relationships among the attributes or fields in the data structure, the database designer can use the normalization process to improve the existing data structure to create an appropriate database design.
- N/B Whether designing a new database structure or modifying an existing one, the normalization process is the same.

The process for normalization:

Each project has its own project number, name, employees assigned to it, and so on. Each employee has an employee number, name, and job classification, such as engineer or computer technician.

The project number (PROJ_NUM) is apparently intended to be a primary key or at least a part of a PK, but it contains nulls. (Given the preceding discussion, you know that PROJ_NUM + EMP_NUM will define each row.)

2. The table entries invite data inconsistencies. For example, the JOB_CLASS value "Elect. Engineer" might be entered as "Elect.Eng." in some cases, "El. Eng." in others, and "EE" in still others.

3. The table displays data redundancies. Those data redundancies yield the following anomalies:

a. *Update anomalies*. Modifying the JOB_CLASS for employee number 105 requires (potentially) many alterations, one for each EMP_NUM = 105.

b. *Insertion anomalies*. Just to complete a row definition, a new employee must be assigned to a project. If the employee is not yet assigned, a phantom project must be created to complete the employee data entry.

c. *Deletion anomalies*. Suppose that only one employee is associated with a given project. If that employee leaves the company and the employee data are deleted, the project information will also be deleted. To prevent the loss of the project information, a fictitious employee must be created just to save the project information.

THE NORMALIZATION PROCESS

The objective of normalization is to ensure that each table conforms to the concept of well-formed relations—that is, tables that have the following characteristics:

- Each table represents a single subject. For example, a course table will contain only data that directly pertain to courses. Similarly, a student table will contain only student data.

- No data item will be unnecessarily stored in more than one table (in short, tables have minimum controlled redundancy). The reason for this requirement is to ensure that the data are updated in only one place.
- All nonprime attributes in a table are dependent on the primary key—the entire primary key and nothing but the primary key. The reason for this requirement is to ensure that the data are uniquely identifiable by a primary key value.
- Each table is void of insertion, update, or deletion anomalies. This is to ensure the integrity and consistency of the data.

To accomplish the objective, the normalization process takes you through the steps that lead to successively higher normal forms.

The concept of keys is central to the discussion of normalization. A **candidate key** is a minimal (irreducible) superkey. The primary key is the candidate key that is selected to be the primary means used to identify the rows in the table.

Functional Dependence

Functional dependence: The attribute B is fully functionally dependent on the attribute A if each value of A determines one and only one value of B.

Functional dependence: Attribute A determines attribute B (that is, B is functionally dependent on A) if all of the rows in the table that agree in value for attribute A also agree in value for attribute B.

Fully functional dependence (composite key): If attribute B is functionally dependent on a composite key A but not on any subset of that composite key, the attribute B is fully functionally dependent on A.

Two types of functional dependencies that are of special interest in normalization are partial dependencies and transitive dependencies.

A **partial dependency** exists when there is a functional dependence in which the determinant is only part of the primary key (remember we are assuming there is only one candidate key).

For example, if (A, B) determines (C,D), B determines C, and (A, B) is the primary key, then the functional dependence B determines C is a partial dependency because only part of the primary key (B) is needed to determine the value of C. Partial dependencies tend to be rather straightforward and easy to identify.

A **transitive dependency** exists when there are functional dependencies such that X determines Y, Y determines Z, and X is the primary key. In that case, the dependency X determines Z is a transitive dependency because X determines the value of Z via Y. Unlike partial dependencies, transitive dependencies are more difficult to identify among a set of data. Fortunately, there is an easier way to identify transitive dependencies. A transitive dependency will occur only when a functional dependence exists among nonprime attributes. In the previous example, the actual transitive dependency is X determines Z.

Conversion to First Normal Form

Repeating group derives its name from the fact that a group of multiple entries of the same type can exist for any *single* key attribute occurrence

A relational table must not contain repeating groups.

Step 1: Eliminate the Repeating Groups

Start by presenting the data in a tabular format, where each cell has a single value and there are no repeating groups. To eliminate the repeating groups, eliminate the nulls by making sure that each repeating group attribute contains an appropriate data value.

Step 2: Identify the Primary Key

Proper primary key that will *uniquely* identify any attribute value

Step 3: Identify All Dependencies

Therefore The term **first normal form** (**1NF**) describes the tabular format in which:

- All of the key attributes are defined.

- There are no repeating groups in the table. In other words, each row/column intersection contains one and only one value, not a set of values.
- All attributes are dependent on the primary key.

Conversion to Second Normal Form

Converting to 2NF is done only when the 1NF has a composite primary key. If the 1NF has a single-attribute primary key, then the table is automatically in 2NF.

Step 1: Make New Tables to Eliminate Partial Dependencies

For each component of the primary key that acts as a determinant in a partial dependency, create a new table with a copy of that component as the primary key. While these components are placed in the new tables, it is important that they also remain in the original table as well. It is important that the determinants remain in the original table because they will be the foreign keys for the relationships that are needed to relate these new tables to the original table.

Step 2: Reassign Corresponding Dependent Attributes

The attributes that are dependent in a partial dependency are removed from the original table and placed in the new table with its determinant. Any attributes that are not dependent in a partial dependency will remain in the original table.

Note: Because a partial dependency can exist only when a table's primary key is composed of several attributes, a table whose primary key consists of only a single attribute is automatically in 2NF once it is in 1NF.

A table is in **second normal form** (**2NF**) when:

- It is in 1NF.

and

- It includes no partial dependencies; that is, no attribute is dependent on only a portion of the primary key.

Note that it is still possible for a table in 2NF to exhibit transitive dependency; that is, the primary key may rely on one or more nonprime attributes to functionally determine other nonprime attributes, as is indicated by a functional dependence among the nonprime attributes.

Conversion to Third Normal Form

Step 1: Make New Tables to Eliminate Transitive Dependencies

For every transitive dependency, write a copy of its determinant as a primary key for a new table. A **determinant** is any attribute whose value determines other values within a row. If you have three different transitive dependencies, you will have three different determinants. As with the conversion to 2NF, it is important that the determinant remain in the original table to serve as a foreign key.

Step 1: Make New Tables to Eliminate Transitive Dependencies

For every transitive dependency, write a copy of its determinant as a primary key for a new table. A **determinant** is any attribute whose value determines other values within a row. If you have three different transitive dependencies, you will have three different determinants. As with the conversion to 2NF, it is important that the determinant remain in the original table to serve as a foreign key.

Step 2: Reassign Corresponding Dependent Attributes

Identify the attributes that are dependent on each determinant identified in Step 1. Place the dependent attributes in the new tables with their determinants and remove them from their original tables.

Using this example

_ The company manages many projects.

_ Each project requires the services of many employees.

_ An employee may be assigned to several different projects.

_ Some employees are not assigned to a project and perform duties not specifically related to a project. Some

employees are part of a labor pool, to be shared by all project teams. For example, the company's executive
secretary would not be assigned to any one particular project.
_ Each employee has a single primary job classification. That job classification determines the hourly billing rate.
_ Many employees can have the same job classification. For example, the company employs more than one
electrical engineer.