Entity-Relationship Modelling & Normalization

Contents

- What is ER Modelling ?
- Building blocks of ER Model
- What is Normalization?
- Normalization Procedure - 1st Normal Form
- Normalization Procedure - 2nd Normal Form
- Normalization Procedure - 3rd Normal Form

What is ER Model?

What is ER Model?

ER Model is a data modelling technique used for describing a database in an abstract way.

The basic building blocks of an ER Model are:

- Entity
- Relationship and cardinality
- Attributes

Entity

An Entity is identified as something which is capable of independent existence and which has a unique identity

An entity can be usually identified by identifying the nouns in a given scenario

Examples

Car, Employee, Book, Course etc.

Relationships and Cardinality

A **Relationship** is the association between two entities in an ER Model.

Example:

Manager supervises an Employee

Librarian issues Books

In the above scenarios, “supervises” and “issues” are the relationships

Cardinality is defined by identifying how many instances of one entity are related to how many instances of another entity at any given time.

The different cardinalities are

- One-to-One
- One-to-Many
- Many-to-One
- Many-to-Many

Attributes

The physical and abstract properties of an Entity are called attributes

Example:

Name is an attribute of the customer entity

Published Year is an attribute of the Book entity

ER Model - Example

Hotel Room Reservation System – A Sample

Entities

Hotel (Hotel_name, address, phone_no, rating)

Guest (Guest_name, address, contact_no)

Room (Room_No, Room_Type, Occupied_Flag, Capacity, Tariff)

Hotel_Staff (Staff_name, address, contact_no, DOB, DOJ, dept, role)

Cardinality

Hotel to Room – 1 to Many (assuming that one room_no can exist only in 1 Hotel)

Hotel to Hotel_Staff – 1 to Many (assuming that the Hotel staff are not transferred to another hotel)

Guest to Room – Many to Many

Guest to Hotel_Staff – Many to Many

What is Normalization?

What is normalization?

Normalization is a process of systematically arranging the data elements, captured as part of executing a business process. The objective of normalization is to avoid or minimize the insert, update and delete anomalies.

Having redundancy is a symptom that a table has anomalies. The process of removing the anomalies also results in a model that has less data redundancy.

Anomalies

Anomaly is a deviation from the normal or the expected standards.

In a database there are 3 types of possible anomalies

- Insert
- Update
- Delete

Insert Anomaly

Consider a database table which stores the details of employees and their departments

Emp_Id	Emp_name	Emp_Desgn	Dept_id	Dept_name	HOD

In the above example, when a new department is inserted, the values for Emp_Id, Emp_name and Emp_Desgn will have to be inserted as null. If you want to avoid null values, the insert has to wait until an Employee joins that department. However, in reality a department exists before an employee joins. This is called as insert anomaly.

Update Anomaly

Consider a database table which stores the details of employees and their departments

Emp_Id	Emp_name	Emp_Desgn	Dept_id	Dept_name	HOD

In the above example, when the HOD of the department changes, all the employee records belonging to that department will have to be updated. This is called update anomaly.

Delete Anomaly

Consider a database table which stores the details of employees and their departments

Emp_Id	Emp_name	Emp_Desgn	Dept_id	Dept_name	HOD

In the above example, when a department no longer exists, it cannot be deleted as if we have to delete it, the employee details of those employees belonging to that department is also lost. This is called delete anomaly.

Normal Forms

There are three basic forms of normalization

- First Normal Form
- Second Normal Form
- Third Normal Form

Normalization Procedure - 1st Normal Form

Employee Project Allocation System

Employee ID	Employee Name	Job Level	Emp Contact	Project ID	Project Name	Project Type	Allocation Type	Client Name	Client Location

<u>1st Normal Form</u>

There should not be any non-atomic values and repeating groups.

- Contact_No can potentially have non-atomic values (attributes that hold multiple values). This has to be split into separate fields as highlighted below.
- Address fields can have multiple lines, for clarity that can also be split into different fields.

Employee_ID, Employee_Name, Job_Level, **Emp_Contact1, Emp_Contact2**, Project_Id, Project_Name, Project_Type, Allocation_type, Client_Name,Client_Location

Normalization Procedure - 1st Normal Form

Next step is to get rid of the repeating groups. The solution is to split the repeating groups into a separate table.

- A project can have many employees assigned to it and an employee can be assigned to more than one project. To avoid details getting repeated, project details are moved to a separate table. Project ID is maintained in the table to show the relationship.

Table T1 (Employee_Allocation):

Project_ID	Employee_ID	Allocation Type	Employee Name	Job Level	Emp_Contact_1	Emp_Contact_2

Table T2 (Project_Details):

Project ID	Project Name	Project Type	Client Name	Client Location

Now the above tables are in 1 NF

Normalization Procedure - 2nd Normal Form

Second Normal Form

The tables are said to be in 2nd Normal form when the following conditions are satisfied

i) The tables should be in 1NF
ii) There should not be any partial dependency.

If there are any non-key attribute which is not dependent on the entire composite key, then that is called **partial dependency**.

The primary key of the table T1 in the above structure is a composite key (combination of more than one field). The composite key consists of Employee ID and Project ID. The non key attributes Employee Name, Job Level and Contact depends only on Employee ID and not on Project_ID. Hence there is partial dependency.

Normalization Procedure - 2nd Normal Form

To remove the partial dependency, the table T1 would be further divided as shown below

Table T1.1 (Employee_Details):

Employee_ID	Employee Name	Job Level	Emp_Contact_1	Emp_Contact_2

Table T1.1 (Project_Allocation_Details):

Project_ID	Employee_ID	Allocation Type

Note: While separating the tables, a foreign key should be maintained to show the relationship.

Normalization Procedure - 3rd Normal Form

For a table to be in **3NF**,

i) it should be in 2NF
ii) there should not be any transitive dependency.

If there is any non key attribute which is dependent on another non key attribute, then **transitive dependency** exists. In other words, all non key attributes should depend on the primary key.

In the above structure,Table 2 has transitive dependency. The Client Name depends on the primary key (Project_ID) whereas Client_Location depends only on Client Name (assuming client names will be unique), and not the primary key(Project_ID). Hence transitive dependency exist.

To remove transitive dependency, client details are moved to a separate table.

Table T2.1 (Project_Details):

Project ID	Project Name	Project Type	Client Name (FK)

Table T2.2 (Client_Details):

Client Name	Client Location

Normalization Procedure - 3rd Normal Form

After 3NF, the final tables would be as shown below

Table T1(Employee_Details):

Employee_ID	Employee Name	Job Level	Emp_Contact_1	Emp_Contact_2

Table T2 (Employee_Allocation_Details):

Project_ID	Employee_ID	Allocation Type

Table T3 (Project_Details):

Project ID	Project Name	Project Type	Client Name

Table T4 (Client_Details):

Client Name	Client Location