Normalization work Document

A close-up of a document

AI-generated content may be incorrect.

A source of data for the database

0NF : is the collection of data from your data source

It is best to look at the labels on your resource document

Employee (**EmployeeID**, EmployeeName, DepartmentNumber, DepartmentName, (ProjectNumber, ProjectName, DepartmentName,WeeklyHours))

Optionally at this point you could name your table. You MAY discover that you might need to rename tables already given a name. This is why this step is optional, and naming of table can wait until the 3NF form is done.

1NF: for an entity to be in this normal form:

1. A table MUST contain atomic attributes. (optionally you **could** atomize in 0NF)
2. A table cannot contain any repeating groups of attributes.
   1. Cut the repeating group from the original entity table AND  
      make it its own table
   2. Copy down the primary key in the original entity table to the new entity table **AND** indicate the attribute as a foreign key
   3. Decide on a primary attribute in your new table that is the unique identifier for the table (primary key for the new table)
3. Optionally at this point you could name your table. You MAY discover that you might need to rename tables already given a name. This is why this step is optional, and naming of table can wait until the 3NF form is done.

Employee (**EmployeeID**, EmployeeFirstName, EmployeeLastName DepartmentNumber, DepartmentName)

Project (**ProjectNumber**, ***EmployeeID***, ProjectName, DepartmentName, WeeklyHours)

With BOTH the projectnumber and employeeid, can I uniquely identify a row in the project table?   
 YES

The result of using two OR more attributes to identify the unique row of the table is called a **composite (compound) key**

**2NF Normal Form : Removing Partial Dependencies**

What is a partial dependency?

A partial dependency exists **ONLY** in tables with compound primary keys.

An attribute is consider to be partially dependent IF it is related on “part of” the compound primary key. Separating out the attribute with the portion of the compound primary key can stand as a table by themselves.

Steps:

1. Examine each non-key attribute against the primary key. IF it can stand by itself along with a portion of the primary key, then form a new table. IF the attribute needs ALL parts of the primary key, then it REMAINS in the table and is NOT a partial dependency.
2. For an attribute that is considered a partial dependency:
   1. Create a new table
   2. Copy the portion of the primary key to the new table and it will become the primary key of the new table.
   3. Move the primary dependency attribute from the original table into the new table.
   4. Make the portion of the primary key in the original table, that was copied to the new table, a foreign key. IT REMAINS A PRIMARY KEY ALSO.
3. Optionally at this point you could name your table. You MAY discover that you might need to rename tables already given a name. This is why this step is optional, and naming of table can wait until the 3NF form is done.

Note: as you proceed through these normalization levels you MAY discover that you need to refine the attribute name to BETTER identify the attribute. Here, it was discovered that DepartmentName in the Project table was confused with the DepartmentName to which the Employee belonged. One solution is to alter the attribute is it better identifies itself. DepartmentName became ProjectDepartmentName.

Employee (**EmployeeID**, EmployeeFirstName, EmployeeLastName DepartmentNumber, DepartmentName)

ProjectHours (***ProjectNumber***, ***EmployeeID***, WeeklyHours)

Project (**ProjectNumber,** ProjectName, ProjectDepartmentName )

An alternative way of noting Primary and Foreign Keys is to use PK and FK

Employee (EmployeeID (PK), EmployeeFirstName, EmployeeLastName DepartmentNumber, DepartmentName)

ProjectHours (ProjectNumber (PK/FK), EmployeeID (PK/FK), WeeklyHours)

Project (ProjectNumber (PK)**,** ProjectName, ProjectDepartmentName )

**3NF Normal Form : Removing Transitive Dependencies**

What is a transitive dependency?

A transitive dependency can exist in any table that has two or more “non-primary key” attributes.

A non-key attribute that is fully dependent on another non-key attribute is termed a **transitive dependency.**

HINT: I use two fingers. One finger will be stationary on an attribute. The other finger moves across the remaining attributes and a decision on dependency is made between the two attributes. So, once you have moved your second finger along the list of all other attributes and reached the end of the attributes, you have checked ALL combinations of the first finger and second finger. Now, move your first finger to the next attribute and repeat the process of comparing the remaining attributes to your new stationary attribute.

Steps:

1. Compare 2 non primary key attributes to together to determine if there is any close association between the attributes. (Is there a “relationship”)
2. For an attribute that is considered a transitive dependency:
   1. Create a new table
   2. Move both attributes to the new table
   3. Determine which of the two attributes would make an appropriate primary key for the new table.
   4. Copy the new table primary key BACK to the original table as ONLY a foreign key.
   5. Continue checking your non-primary key attribute combinations, any additional relationship to the NEW foreign key in the original table moves the non-foreign key attribute to the new table.
3. Continue examining each of your existing tables. You MAY discover that another table has an attribute that is extremely similar to your new table attributes. If this exists, consider changing the attribute to a foreign key using the new table primary key attribute. In this example the Project ProjectDepartment COULD (and should) be replaced with the primary key attribute of the new table as a foreign key.
4. Optionally at this point you could name your table. You MAY discover that you might need to rename tables already given a name. This is why this step is optional, and naming of table can wait until the 3NF form is done.

Employee (**EmployeeID**, EmployeeFirstName, EmployeeLastName, *DepartmentNumber*)

ProjectHours (***ProjectNumber***, ***EmployeeID***, WeeklyHours)

Project (**ProjectNumber,** ProjectName, *DepartmentNumber* )

Department (**DepartmentNumber**, DepartmentName)

**Merging Views into one set of tables**

The final step in Normalization, would be to take all the separate Views and creates a single set of tables that represent the database for the work.

Steps:

1. Take two views and compare the set of tables within the views.
2. If there is two tables with the same Primary Key, then these two table need to be combined into a single table
   1. Choose one of the tables as the final combined table
   2. In the second table, compare each attribute to the final combined table.
   3. IF the attribute does NOT exist in the combined table, place a copy of the attribute in the combined table.
   4. Repeat for each attribute in the second view table
3. Repeat step 2 for all views of your normalization process