Database Normalization

Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more consistent by eliminating redundancy and inconsistent dependency.

Redundant data wastes disk space and creates maintenance issues. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations. A student address change is much easier to implement if that data is stored only in the Student table and nowhere else in the database.

There are a few rules for database normalization. Each rule is called a "normal form." If the first rule is observed, the database is said to be in "first normal form." If the first three rules are observed, the database is considered to be in "third normal form." Although other levels of normalization are possible, third normal form is considered the highest level necessary for most applications.

First Normal Form

- Eliminate repeating groups in individual tables.
- Create a separate table for each set of related data.
- Identify each set of related data with a primary key.

Do not use multiple fields in a single table to store similar data. For example, to record for a class that may have two possible books, an inventory record might contain fields for ISBN1 and ISBN2.

What happens when you add a third book? Adding a field is not the answer; it requires table modifications and does not smoothly accommodate a dynamic number of books. Instead, place all book information in a separate table called Books, then link classes to books with a book number key as an individual row.

Second Normal Form

- Create separate tables for sets of values that apply to multiple records.
- Relate these tables with a foreign key.

Records should not depend on anything other than a table's primary key (a compound key, if necessary). For example, consider a student's address in a registration system. The Student table has the address, but other tables such as Billing, Registration, and Grades also require the address. Instead of storing the student's address as a separate entry in each of these tables, store it in one place, either in the Student table or in a separate Address table.

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.

Think of items such as a Zip Code in all the street addresses. In theory, normalization is worth pursing. However, many small tables may degrade performance or exceed open file and memory capacities. It may be more feasible to apply third normal form only to data that changes frequently.

Normalizing an Example Table

These steps demonstrate the process of normalizing a fictitious student table.

1. Unnormalized table:

StudentID	Advisor	Office	Class1	Class2	Class3
a1001	Miller	4242	12345	67890	54321
b2002	Martinez	4241	44556	22339	77665

2. First Normal Form: No Repeating Groups

Tables should have only two dimensions. Since one student has several classes, these classes should be listed in a separate table. Fields Class1, Class2, and Class3 in the above records are indications of design trouble.

Spreadsheets often use the third dimension, but tables should not. Another way to look at this problem is with a one-to-many relationship, do not put the one side and the many side in the same table. Instead, create another table in first normal form by eliminating the repeating group (Class#), as shown below:

StudentID	Advisor	Office	Synonym
a1001	Miller	4242	12345
a1001	Miller	4242	67890
a1001	Miller	4242	54321

StudentID	Advisor	Office	Synonym
b2002	Martinez	4241	44556
b2002	Martinez	4241	22339
b2002	Martinez	4241	77665

3. Second Normal Form: Eliminate Redundant Data

Note the multiple Synonym values for each StudentID value in the above table. Synonym is not functionally dependent on StudentID (primary key), so this relationship is not in second normal form.

The following two tables demonstrate second normal form:

Students:

StudentID	Advisor	Office
a1001	Miller	4242
b2002	Martinez	4241

Registration:

StudentID	Synonym
a1001	12345
a1001	67890
a1001	54321
b2002	44556
b2002	22339
b2002	77665

4. Third Normal Form: Eliminate Data Not Dependent On Key

In the last example, Office (the advisor's office number) is functionally dependent on the Advisor attribute. The solution is to move that attribute from the Students table to the Faculty table, as shown below. This is one of many possible solutions which are dependent on the performance and usability of the data in the database.

Students:

StudentID	Teacher
a1001	Miller
b2002	Martinez

Faculty:

Name	Room	Department
Miller	4242	Computer Science
Martinez	4241	Computer Science

Registration:

StudentID	Synonym
a1001	12345
a1001	67890
a1001	54321
b2002	44556
b2002	22339
b2002	77665