Normalization

Introduction to 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 flexible by eliminating **redundancy** and **inconsistent dependency**.

Normalization

Introduction to normalization: eliminating redundancy

- Redundant data wastes disk space and creates maintenance problems.
- 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 customer address change is much easier to implement if that data is stored only in the Customers table and nowhere else in the database.

Normalization

Introduction to normalization: inconsistent dependency

- What is an "inconsistent dependency"? While it is intuitive for a user to look in the Customers table for the address of a particular customer, it may not make sense to look there for the salary of the employee who calls on that customer. The employee's salary is related to, or dependent on, the employee and thus should be moved to the Employees table.

- Inconsistent dependencies can make data difficult to access because the path to find the data may be missing or broken.

Normalization

Rules for normalization

- There are a few rules for database normalization. These rules are proven and are based on database design theory.

- 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.

- As with many formal rules and specifications, real world scenarios do not always allow for perfect compliance. In general, normalization requires additional tables and some developers find this cumbersome.

Normalization

Rules for normalization: First Normal Form (1NF)

- 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 (covered next).

Do not use multiple fields in a single table to store similar data. For example, to track an inventory item that may come from two possible sources, an inventory record may contain fields for Vendor Code 1 and Vendor Code 2.

What happens when you add a third vendor? Adding a field is not the answer; it requires program and table modifications and does not smoothly accommodate a dynamic number of vendors. Instead, place all vendor information in a separate table called Vendors, then link inventory to vendors with an item number key, or vendors to inventory with a vendor code key.

Normalization

Rules for normalization: Second Normal Form (2NF)

- 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. For example, consider a customer's address in an accounting system. The address is needed by the Customers table, but also by the Orders, Shipping, Invoices, Accounts Receivable, and Collections tables. Instead of storing the customer's address as a separate entry in each of these tables, store it in one place, either in the Customers table or in a separate Addresses table.

Normalization

Rules for normalization: Third Normal Form (3NF)

- 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.

In an Employee Recruitment table for instance, a candidate's university name and address may be included. But you need a complete list of universities for group mailings. If university information is stored in the Candidates table, there is no way to list universities with no current candidates. Create a separate Universities table and link it to the Candidates table with a university code key.

Note: Adhering to the third normal form, while theoretically desirable, is not always practical. In theory, normalization is worth pursing. However, many small tables may ultimately degrade performance or exceed open file and memory capacities.

Normalization

An example of normalization

In the next few slides, we'll use a fictitious Students table to demonstrate normalization. Consider the following unnormalized table:

studentid	advisor	advroom	class1	class2	class3
101	Sally Smith	412	10107	14301	15902
102	Mark Jones	216	20101	21102	21401

Normalization

An example of normalization

First Normal Form: No Repeating Groups

Since one student has several classes, these classes should be listed in a separate table. Fields class1, class2, and class3 are indications of design trouble. The previous table could be normalized as follows:

studentid	advisor	advroom	classnumber
101	Sally Smith	412	10107
101	Sally Smith	412	14301
101	Sally Smith	412	15902
102	Mark Jones	216	20101
102	Mark Jones	216	21102
102	Mark Jones	216	21401

Of course the problem here is glaring: we now have redundant data…

Normalization

An example of normalization

Second Normal Form: Eliminate Redundant Data

Note the multiple classnumber values for each studentid value in the previous table. classnumber is not functionally dependent on studentid, so this relationship is not in second normal form. The following two tables demonstrate a better design (second normal form):

studentid	advisor	advroom
101	Sally Smith	412
102	Mark Jones	216

studentid	classnumber
101	10107
101	14301
101	15902
102	20101
102	21102
102	21401

Normalization

An example of normalization

Third Normal Form: Eliminate Data Not Dependent On Key

In the last example, advroom (the advisor's office number) is functionally dependent on the advisor. The solution is to move that field from the Students table and into a new table called Faculty, as shown below:

studentid	advisor
101	Sally Smith
102	Mark Jones

ılty

advisor	room	department
Sally Smith	412	42
Mark Jones	216	42
Dave Davis	111	42
Todd Sing	520	42