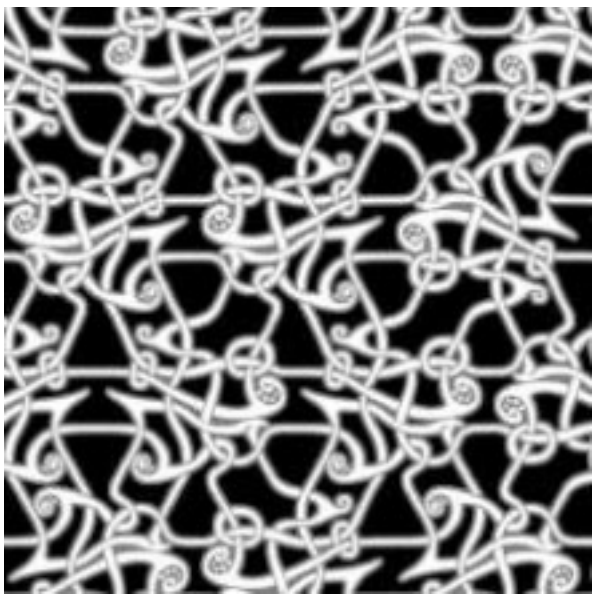
Database Normalization Explained in Simple English

Introduction to Database Normalization



Database normalization is process used to organize a database into tables and columns. The idea is that a table should be about a *specific* topic and that only those columns which support that topic are included. For example, a spreadsheet containing information about sales people and customers serves several purposes:

- Identify sales people in your organization
- List all customers your company calls upon to sell product
- Identify which sales people call on specific customers.

By limiting a table to one purpose you reduce the number of duplicate data that is contained within your database, which helps eliminate some issues stemming from database modifications. To assist in achieving these objectives, some rules for database table organization have been developed. The stages of organization are called normal forms; there are three normal forms most databases adhere to using. As tables satisfy each successive database normalization form, they become less prone to database modification anomalies and more focused toward a sole purpose or topic. Before we move on be sure you understand the definition of a database table.

Reasons for Database Normalization

There are three main reasons to normalize a database. The first is to minimize duplicate data, the second is to minimize or avoid data modification issues, and the third is to simplify queries. As we go through the various states of normalization we'll discuss how each form addresses these issues, but to start, let's look at some data which hasn't been normalized and discuss some potential pitfalls. Once these are understood, I think you'll better appreciate the reason to normalize the data. Consider the following table:

SalesStaff						
EmployeeID	**SalesPerson**	**SalesOffice**	**OfficeNumber**	**Customer1**	**Customer2**	**Customer3**
1003	Mary Smith	Chicago	312-555-1212	Ford	GM	
1004	John Hunt	New York	212-555-1212	Dell	HP	Apple
1005	Martin Hap	Chicago	312-555-1212	Boeing		

Note: The primary key columns are underlined

The first thing to notice is this table serves many purposes including:

1. Identifying the organization's salespeople
2. Listing the sales offices and phone numbers
3. Associating a salesperson with an sales office
4. Showing each salesperson's customers

As a DBA this raises a red flag. In general I like to see tables that have one purpose. Having the table serve many purposes introduces many of the challenges; namely, data duplication, data update issues, and increased effort to query data.

Data Duplication and Modification Anomalies

Notice that for each SalesPerson we have listed both the SalesOffice and OfficeNumber. This information is duplicated for each SalesPerson. Duplicated information presents two problems:

1. It increases storage and decrease performance.

2. It becomes more difficult to maintain data changes.

For example

- Consider if we move the Chicago office to Evanston, IL. To properly reflect this in our table, we need to update the entries for all the SalesPersons currently in Chicago. Our table is a small example, but you can see if it were larger, that potentially this could involve hundreds of updates.

These situations are modification anomalies and can be fixed with database normalization. There are three modification anomalies that can occur:

Insert Anomaly

EmployeeID	SalesPerson	SalesOffice	OfficeNumber	Customer1	Customer2	Customer3
1003	Mary Smith	Chicago	312-555-1212	Ford	GM	
1004	John Hunt	New York	212-555-1212	Dell	HP	Apple
1005	Martin Hap	Chicago	312-555-1212	Boeing		
???	???	Atlanta	312-555-1212			

There are facts we cannot record until we know information for the entire row. In our example we cannot record a new sales office until we also know the sales person. Why? Because in order to create the record, we need provide a primary key. In our case this is the EmployeeID.

Update Anomaly

EmployeeID	SalesPerson	SalesOffice	OfficeNumber	Customer1	Customer2	Customer3
1003	Mary Smith	Chicago	312-555-1212	Ford	GM	
1004	John Hunt	New York	212-555-1212	Dell	HP	Apple
1005	Martin Hap	Chicago	312-555-1212	Boeing		

The same information is recorded in multiple rows. For instance if the office number changes, then there are multiple updates that need to be made. If these updates are not successfully completed across all rows, then an inconsistency occurs.

Deletion Anomaly

EmployeeID	SalesPerson	SalesOffice	OfficeNumber	Customer1	Customer2	Customer3
1003	Mary Smith	Chicago	312-555-1212	Ford	GM	
~~1004~~	~~John Hunt~~	~~New York~~	~~212-555-1212~~	~~Dell~~	~~HP~~	~~Apple~~
1005	Martin Hap	Chicago	312-555-1212	Boeing		

Deletion of a row can cause more than one set of facts to be removed. For instance, if John Hunt retires, then deleting that row cause us to lose information about the New York office.

Search and Sort Issues

The last reason we'll consider is making it easier to search and sort your data. In the SalesStaff table if you want to search for a specific customer such as Ford, you would have to write a query like

```
SELECT SalesOffice
FROM SalesStaff
WHERE Customer1 = 'Ford' OR
      Customer2 = 'Ford' OR
      Customer3 = 'Ford'
```

Clearly if the customer were somehow in one column our query would be simpler. Also, consider if you want to run a query and sort by customer. The way the table is currently defined, this isn't possible, unless you use three separate queries with a UNION. These anomalies can be eliminated or reduced by properly separating the data into different tables, to house the data in tables which serve a single purpose. The process to do this is called database normalization, and the various stages you can achieve are called the normal forms.

Database Normalization Simplified.

Definition of Database Normalization

There are three common forms of database normalization: 1st, 2nd, and 3rd normal form. There are several additional forms, such as BCNF, but I consider those advanced, and not too necessary to learn in the beginning. The forms are progressive, meaning that to qualify for 3rd normal form a table must first satisfy the rules for 2nd normal form, and 2nd normal form must adhere to those for 1st normal form. Before we discuss the various forms and rules in detail, let's summarize the various forms:

- **First Normal Form** – The information is stored in a relational table and each column contains atomic values, and there are not repeating groups of columns.
- **Second Normal Form** – The table is in first normal form and all the columns depend on the table's primary key.
- **Third Normal Form** – the table is in second normal form and all of its columns are not transitively dependent on the primary key

Do not get too hung up if you don't know what these rules mean at the moment; we'll explain them in detail in the next post using examples.

For now it's important to understand there are three rules for database normalization which build upon each other. Some people make database normalization seem complicated, but it doesn't have to be, and once you understand it, it becomes intuitive.

More tutorials are to follow! Remember! I want to remind you all that if you have other questions you want answered, then post a comment or tweet me. I'm here to help you. What other topics would you like to know more about?