Database Design Tips - Michael J. Hernandez

Note: This article first appeared in Database Web Advisor magazine. Their version was edited to fit the page layout they had planned and unfortunately, they edited out crucial components. Here is the original text, complete and properly edited, as it was intended.

Using a database nowadays is – for the most part – quite easy. With the many tools available in most Relational Database Management System (RDBMS) programs, retrieving the information you need takes nothing more than building the right query. Seeing the information you need on forms and reports has become a far more manageable task as well. But the question that comes to mind is this: Are you *really* getting the information you need? Is it as accurate as possible? Is it timely?

In order to meet your information requirements, your database must be structured in such a way that it stores the data necessary to support that information in as an effective and efficient manner as possible. It should guarantee the consistency and validity of the data, and it should support every level of data integrity. How do you achieve this seemingly lofty goal? You go through the process of *logical database design*.

You probably have a good idea of how a database should be designed, and you may even be familiar with a design methodology of some sort. The problem is that it can be difficult to stick to the design process, given the types of time constraints that may be imposed upon you. However, sticking to the design process as much as possible will serve you well, because in the end you will have a blueprint from which to work when you implement the database in your RDBMS program. To that end, I've provided a few tips for you to keep in mind while you're working on developing the structure of your database.

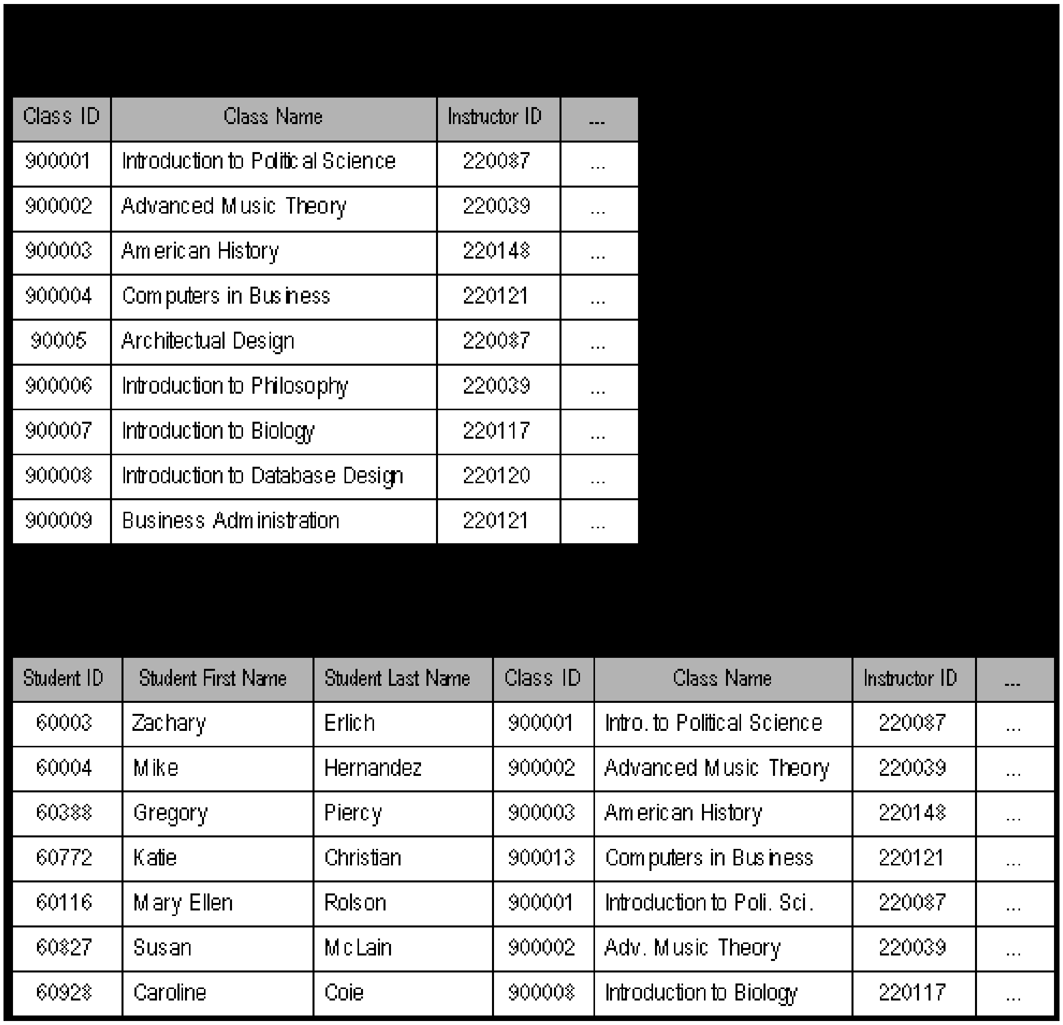
Always design the database on paper first. This one tip will probably save you an enormous amount of time. Designing the database on paper first guarantees that you will keep focused on the task of designing the database as effectively and efficiently as possible. You want to make certain that your database will have only a minimum amount of redundant data and that you've eliminated as much duplicate data as possible. Making changes to the structure is also a much simpler task if done on paper. You'd be surprised how fast an eraser works.

Many people make the mistake of designing a database with their particular RDBMS in mind and will design tables in a manner they hope will fit certain forms or reports they envision themselves using. In most cases, designing a table in this manner will result in poorly constructed tables. Common side effects of this type of construction include difficulty in sorting the data and searching for specific values. In many cases, the tables will have to be redesigned using proper database design techniques.

Don't use an existing database structure as the basis for a new database structure. Your current database has problems – inconsistent data, redundant data, and inaccurate information. Because it's been around for a while, you decide to circumvent these problems by creating a new database from scratch; you figure you can create more efficient structures and address some of the new information requirements that have recently come to light. As you work on the new structure, you figure that there are parts of the existing database that you could probably use. However strong your urge to do so, fight it. The main problem with using the existing database structure is that you may import errors into the new structure that are difficult to identify. You may be inadvertently importing awkward table structures, poorly defined relationships, or inconsistent field definitions. And as always, these problems won't surface until the most inopportune time or until it's too late to do much about them.

Make absolutely certain that each table only represents one subject. A subject can be one of two things: and *object* or an *event*. In this case, and object represents something tangible – such as a person, place, or thing – while an event represents something that occurs at a specific point in time. Both have characteristics that can be stored as data and processed later as information.

When a table describes more than one subject, you will certainly have unnecessary duplicate data and possibly redundant data as well. Ensuring that a table represents a single subject guarantees that you will avoid these problems. For example, consider the following tables:



Class ID	Class Name	Instructor ID	...
900001	Introduction to Political Science	220087	...
900002	Advanced Music Theory	220039	...
900003	American History	220148	...
900004	Computers in Business	220121	...
90005	Architectual Design	220087	...
900006	Introduction to Philosophy	220039	...
900007	Introduction to Biology	220117	...
900008	Introduction to Database Design	220120	...
900009	Business Administration	220121	...

Student ID	Student First Name	Student Last Name	Class ID	Class Name	Instructor ID	...
60003	Zachary	Erlich	900001	Intro. to Political Science	220087	...
60004	Mike	Hernandez	900002	Advanced Music Theory	220039	...
60388	Gregory	Piercy	900003	American History	220148	...
60772	Katie	Christian	900013	Computers in Business	220121	...
60116	Mary Ellen	Rolson	900001	Introduction to Poli. Sci.	220087	...
60827	Susan	McLain	900002	Adv. Music Theory	220039	...
60928	Caroline	Coie	900008	Introduction to Biology	220117	...

As you can see, the Students table clearly represents more than one subject. The immediate problem is that you have duplicate data, which you will need to maintain. If you modify a Class Name in the Students table, you must be sure to make the same modification to the same Class Name in the Classes table. Having structures of this sort also results in another serious problem: inconsistent data. Take a look at the "Introduction to Political Science" class in both tables to see what I mean. Just remember: One Table – One Subject.

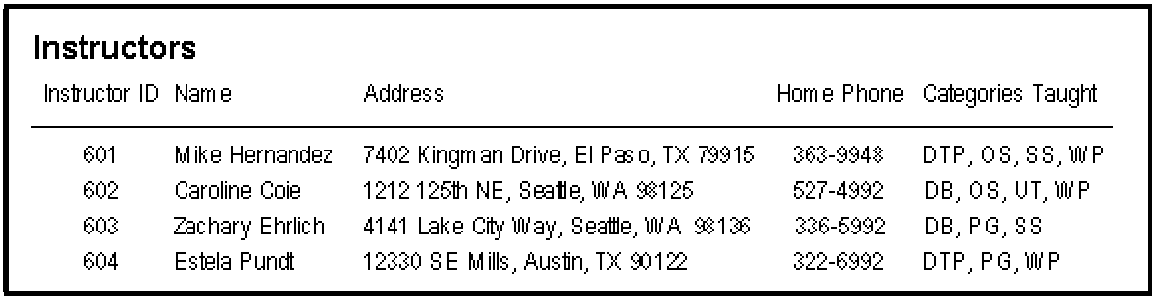
Every table should have a Primary Key. This is an important tip for two very good reasons. First, a Primary Key uniquely identifies each record in a given table and helps to ensure against redundant data. This is the mechanism used to refer to a particular record from other tables in the database. Second, it is also the instrument used to establish a relationship between a pair of tables. This will be of particular importance when you want to retrieve data from multiple tables in a query.

Here are some guidelines for establishing a Primary Key:

- It’s value must be unique.
- It can never be null.
- It cannot be a multi-part field.
- It should comprise the minimum number of fields to guarantee uniqueness.
- It is not optional in whole or in part.
- It must directly identify each value of the remaining fields in a given record of a table.

In order for a field to become a Primary Key for its parent table, it must pass each of these guidelines. Failure to pass even one will disqualify it as a possible Primary Key. If you make certain you have established a valid Primary Key, you will greatly reduce the possibility of encountering problems later when you begin to work with table relationships.

A table should not have any multi-part or multi-valued fields. Let me begin by stating that a *multi-part* field is a field whose value can be divided into smaller parts, while a *multi-valued* field contains multiple occurrences of the same type of value. Consider the following example:



Instructors

Instructor ID	Name	Address	Home Phone	Categories Taught
601	Mike Hernandez	7402 Kingman Drive, El Paso, TX 79915	363-9948	DTP, OS, SS, WP
602	Caroline Coie	1212 125th NE, Seattle, WA 98125	527-4992	DB, OS, UT, WP
603	Zachary Ehrlich	4141 Lake City Way, Seattle, WA 98136	336-5992	DB, PG, SS
604	Estela Pundt	12330 SE Mills, Austin, TX 90122	322-6992	DTP, PG, WP

As you can see, you have two multi-part fields (Name and Address) and one multi-valued field (Categories Taught). Imagine how difficult it would be to sort this data by last name, to retrieve information for everyone living in Washington State, or to find out who teaches Word Processing (WP) classes. This is a classic example that illustrates why you want to avoid these types of fields. The way to resolve these problems is to break the multi-part fields into separate and distinct fields, and build a new table called Instructor Categories that contains the following fields: Instructor ID and Category. In this way, an instructor can be associated with any number of categories.

Invest the time to implement data integrity. I can't over-emphasize the importance of this tip. Many of the problems you'll encounter with inaccurate or erroneous information will be a direct result of poor data integrity. While it sometimes seems like a waste of time to pay so much attention to the many details involved in establishing data integrity, it will actually save you an enormous amount of time in the long run – you won't have to continually go back to fix things. An interesting fact is that the very people who "…just don't have the time." to establish proper data integrity are the ones who usually spend a large amount of time fixing their improperly designed databases. In many cases they will spend up to three times the amount of time it would have taken to design the database properly in the first place! So don't do it over – do it right!

I hope these tips will help you keep on track when you're designing your database. Just remember not to rush through the process. Take your time, keep focused, and design your database as thoroughly as possible before you implement it in your RDBMS. If you do this, you'll have an excellent blueprint from which to work.