Software Engineering & the Basics of Good Database Design

Prepared by Michael A. Stratton, Software Analyst
"If you can dream it, I can build it"
http://www.mikestratton.net
mike@mikestratton.net
Mobile (330)802-0285

Software Engineering & the Basics of Database Design

What is database design? How does one go about effectively administering a database? How does database design relate to the creation, maintenance and scalability of software? This article will answer those questions and more, in an effort to help you understand the software development process.

Basics of Database Design

Although this article does briefly touch on the software development process, the main focus is the basics of database design, a “sub-set” of software engineering.

This article focuses on a few basic concepts, such as:

- Normalization
- Maintenance
- Relational Databases
- Performance
- Scalability
- Good Database Design Methods

Normalization: The process of structuring data in order to minimize duplication and inconsistencies.

Maintenance: A database should be easy to maintain. Do not use repetitive data. This can be accomplished by creating a table (master table) with all possible values, and then using a key to refer to the value.

For example, imagine that you are responsible for managing a database of customers and the products they have purchased. If 5,000 of these customers have purchased a product called “flashy widget”, this name would appear 5,000 times in the customer table. If the name of the product is changed to “red widget, this name would need to be changed 5,000 times. If the database was designed so that product names appeared in one table, and just the product ID number was stored with the customer name, you would only have to change the product name 1 time, rather than 5,000.

The worst possible time to redesign a database is immediately after the public launch of a software product. Before you start coding, spend a lot of time designing your database.

Three Types of Table Relationships

- One to One
- One to Many
- Many to Many

Let's use a table called customer and a table called PurchaseDetails as an example to show the relationship of the customer and the store(s) in which they purchase products. The key in the tables represent the Primary Key.

Customer		
SSN	CustomerName	PurchaseDetails
555-55-xxxx	John Smith	66188
888-88-xxxx	Jane Smith	44433, 22220
123-12-xxxx	Larry Brown	22220
11-111-xxxx	Henry Lanceworth	66188

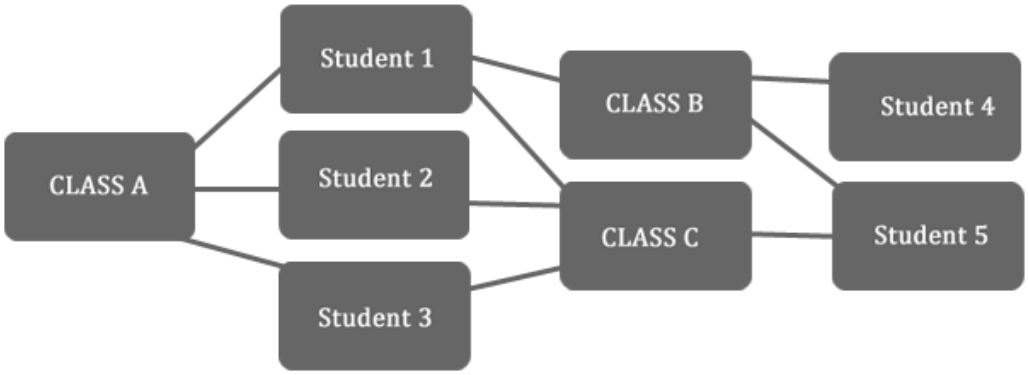
PurchaseDetails	
StoreID	Store Location
66188	New York, NY
44433	Dallas, TX
22220	Online Website

The above is an example of a one-to-many relationship, as more than one customer can purchase from a particular store. The purchase details on the other hand, is created for the sole purpose of designating store locations of purchases and does not store Primary Key's from other tables.

DriversLicence			
CardID	SSN	AmountSpent	CardNumber
010203	555-55-xxxx	$27.14	123567
220011	888-88-xxxx	$365.29	555333
123123	123-12-xxxx	$98.52	222222
555555	111-11-xxxx	$55.66	888999

The driver's license and customer table relationship is an example of a one-to-one relationship as a customer can only have one driver's license.

The best way to explain and design a many to many relationship within a database is to think in terms of a one-to-many relationship even though a many-to-many relationship exists.



For example, a student can take multiple classes whereas a class can hold multiple students; therefore, a class can hold multiple instances of studentID's and a student can hold multiple instances of classID's.

Class	
classID	Name
1234	English II
2345	Algebra
3456	Economics

Student	
studentID	Name
1011	Colt McDonald
1012	Erin Rogers
1013	Barbara Woodard
1014	Jonathon Nevins
1015	Bridget Johnson

To help in the design of a many-to-many relationship, you could create an intermediate table, one that maps the tables together.

Normalization

To help insure that your database design is efficient and easy to maintain, it helps to think of normalization as a set of rules to follow, known as normal forms. There are three steps to follow in the process of normalization; a) first normal form, b) second normal form, and c) third normal form.

Please note: Although this article uses three normal forms as a standard, other uses of normalization exist; such as, Boyce-Codd normal form, fourth normal form, and fifth normal form/Join- Projection normal form. Although these methods of normalization have their benefit, they are not recommended as a current standard.

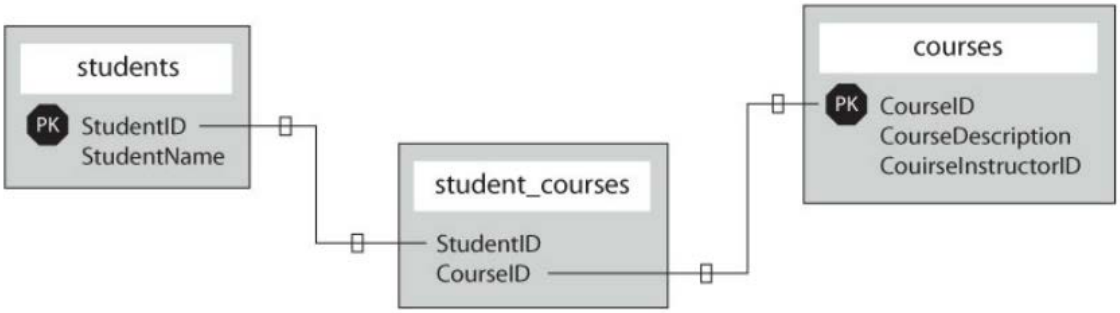
First Normal Form

- Eliminate repeating data
- Create separate tables for related data

Second Normal Form

- No non-key attributes depend on a portion of the primary key.

In other words, the fields in your table should be entirely related to only one primary key. Using the students and courses tables as an example, we improve the efficiency of the database. Without this map, the students table relies on the StudentID primary key AND the ClassID primary key. A new table, StudentCourses, resolves this problem in efficiency.



Third Normal Form

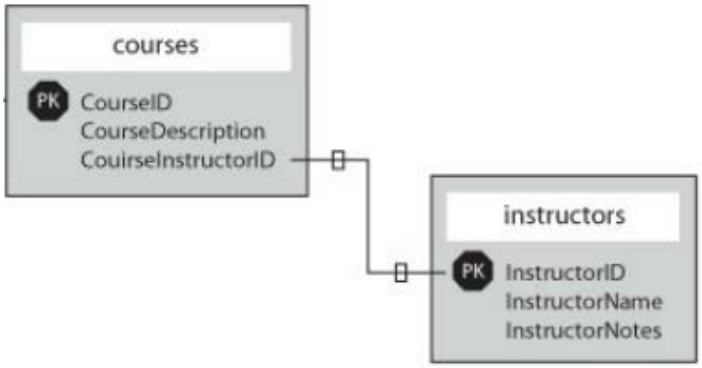
- No attributes depend on other non-key attributes.

This means that you should review your tables to check for fields that can be broken down further that are not dependent on a primary key.

For example, the following table should be split into two tables.

Courses
CourseID
CourseDescription
CourseInstructorID
Instructors

We derive two tables, courses and instructors, and then connect the two via the primary key, instructorID.



The third normal form removes redundancy while improving extensibility.

The Design Process

A continuing problem in the design, development, and maintenance of applications is the lack in planning. The design process must include an extensive evaluation of the database. Some things to consider while designing your database are as follows. What should your database hold? How does the data relate to other data? It is also imperative that your database is scalable. Is your database scalable?

Generally, the design process of a database can be broken down into steps.

- Define objective of the database
- Design structures (tables, fields)
- Define, evaluate, and re-evaluate relationships
- Define and implement business rules
- Develop/code the application

Please note: Some software analysts take an agile approach in the development of software and will spend little or no time in the design stage, moving directly into the coding stage. The argument that exists supporting this methodology states that this improves on the timeframe, usability and performance as a result of the stages being iterative (repetitive) as opposed to being incremental. This article was written from a non-agile perspective; yet, I do not agree or disagree with the agile approach. I believe that agile software development can be an extremely useful software development methodology, especially when there are extreme time constraints. My view is that both the traditional (waterfall methodology) and modern (agile methodology) have their place.

Summary

In the same manor that a mechanical aircraft engineer plans analyzes, designs and creates the blueprints for an airplane, a software analyst/engineer must analyze, design and code by a set of industry standards.

Complete comprehension of relational databases is essential in the development of any systems application, as is the ability to evaluate and choose the proper methodology for a specific software project

Database design is one small portion of the creation of robust, performance-driven, extensible software. Whether using the waterfall, agile or other approach when creating software, it is imperative that a software engineer focus on functionality, maintenance and scalability.

---

http://www.mikestratton.net
mike@mikestratton.net
Mobile (330)802-0285

References

Element K. (Unknown). *SAMS Teach Yourself PHP, MySQL and Apache All in One [2nd Edition]*. Retrieved July 3, 2012, from Element K Corporation: http://www.elementk.com

Comment: *The online course, "SAMS Teach Yourself PHP, MySQL and Apache All in One [2nd Edition]", was provided by the Association for Computing Machinery, in partnership with Element K Corporation*.

Association for Computing Machinery (2012), http://www.acm.org