Jacobi

Finally, got a chance to read the article on basic database design.  I truly found it to be a good illustration of what a good basic database needs to look like.  From organizing information into tables divided up by list or rows (record) and columns(field). To understanding the 3 out of the 5 normal forms.  Great read for beginners and not difficult to understand which i feel is very important in learning this new process of thinking.

Lets start with tables, columns, and rows because they are the ones that have been stomping me only because of the difficulty on identifying which one is which.  After reading this article I have better understanding on how to recognize what information goes where.

Never knew how detailed and organized you have to be in order to get your code to do exactly what you want it to do.  The skill of being able to analyze properly will be greatly used in knowing what you need aka knowing the purpose of the database.

For instance, the table will store the information of the column (field) normally is one word category to make the column unique.  The column hold the records or results from what is called from the table. Then finally the record is displayed once it is called up properly from each column.

Tables                          Columns                          row

Customers                   First Name                      Steve

                                                                            Bob

                                                                            Jim

                                    Last Name                      Jenkins

                                                                           Wallace

                                                                           Hennessey

                                   city, state, zip                  Tampa, fl, 33511

                                                                           little elm, tx, 75068

                                                                           frisco, tx, 75688

                                  email                                stevej@gmail.com

                                                                          bwallace@gmail.com

                                                                         Hennjimbo@gmail.com

Dog/Pup                    Dog/Pup ID                    1154476

                                                                         8877665

                                                                         9809765

                                  breed                              American Bully

                                                                         French Bulldog

                                                                         Pitbull

                                  male                               y

                                                                         n

                                                                         y

                                 female                             n

                                                                         y

                                                                         n

What i also learned  here is that primary keys can link to another table by a foreign so that table can use the value.  By using a foreign key they become primary key to the junction table.  I hope I said that right!  LOL!

Here is another important fact that i pounded in my head was the fact that we do not want to state a fact more than once.  Program to where you can use it whenever you want to off just one line of code.

In the worlds table on SQL i saw country and country languages which i would have put country language in one of the rows in the country tab.  Yes, people speak the same language in different places but you can still narrow it down to get any answer you need.

Also, in the city columns it has ID and Name.  I just think it should be one or the other because yes their are same cities in different countries.  The field Country code exist so that would separate it.  That just my opinion.

To sum it all up I see the database is mainly a table made up of columns and rows.  The database needs to be called up from the user side to retrieve some information about the topic.  Practicing good database basics will make my life a lot easier and who ever else I may be working with.

--

Brian Jenkins



What is good database design? Good database design should divide your information into subject-based tables that reduce redundant data, help support the accuracy and integrity of your information, and accommodate your data processing and reporting needs.  After reading Microsoft’s article on database design basics, I will discuss the main points I took from the article, what questions you should ask when creating a database, how to determine the key of your table, and give you some examples from the sakila and world databases.

            Some important points that I picked up from this article were to plan and ask yourself the logical questions with regards to how it will be utilized present and future. It is important to reduce redundant information, both to maximize your disk space and to ensure accurate information. It is important to understand the different relationships that tables can have between each other and how to determine when to use each type. One-to-one relationships are typically used for information that is rare or only applies to a few records that can be placed on a separate table. One-to-many relationships joins related tables by establishing pairings of primary and foreign keys. Lastly, many-to-many relationships require the use of a third junction table to break down the relationship into two one-to-many relationships.

            How do you determine the purpose of your database and what questions should you ask yourself when creating one? First you should think about what you might want the database to answer. “It makes good sense to construct a prototype of each report or output listing and consider what items you will need…” Consider what type of business will be using the database, how many people will be using it, and what they will use it for. If your database consists of purchase orders, products, vendors, and customers then you might want to ask yourself how many tables and fields you will need. You should note specific reports that someone might pull and create test data to mimic those reports. You should also think about what types of information your data might hold since this will help determine how many fields or tables you will need. For example, if you have a customer information card, what kind of details would be listed on there? All those details would represent a new field within the database.

            Determining the keys plays a key role in building your database. You should determine if there is an existing portion of the information that can be used as the key or if you will have to create a unique key for each record. The primary key must have a value, it cannot be null, and there cannot be duplicates. You can group two columns of information in a table to create a unique primary key, also known as a composite key. You should also consider if the information you want to use is factual or fact-less. Using information like a telephone number or an address, when the information changes it gives room for errors if the key is changed in one table but not another.

            One example of a many-to-many relationship in the Sakila database is the relationship between the film and actor tables since a film can contain several different actors and an actor can star in several different films. A solution for this problem could be to create a junction table to create two one-to-many relationships between the tables. Another example from this database would be the relationship between the film and categories (genre) tables. This would fall into the same problem as the others, some films can be categorized in more than one genre and each genre can contain more than one film.

            An observation I made in the world database was that the city and country tables both contained a population data field. This creates redundant information which can lead to slower reporting. Since each the tables are already connected with a foreign key, a solution for this would be to remove this information from the country table. Rather than calculating the data from each city and populating it into the country table, you should input this calculation step into your query.

            In conclusion, it is important to understand what makes a good database. Ask yourself questions to determine how to organize your database, to understand the importance of determining keys for your tables, and to understand the difference between each table relationship and when you should use them. You should create your database with the data accuracy, integrity, and efficiency in mind.

Brian Johnson



The article “Database design basics” was an interesting read, and pointed out a lot of good facts that every beginner should know when thinking about database design.  I was reassured to know that some of the things that I have been taking into consideration when designing the few databases that I have designed to this point in my professional career were some of the recommendations that were contained in this article.

                The most important concept of this article in my opinion is understanding what questions the database should be able to answer.  In my professional experience with designing databases, though very minimal is all about answering questions from the data.  The database and the tables and data stored within them needs to be able to answer the questions that the business needs to ask of it.  The data itself will be used to build reports, and ad hoc queries that the business will use in day to day operations, and if the database is not structured in a way that supports answering the questions that the business needs to ask then it is useless.  It also should be designed in a fashion that would make it easily modifiable to change and adapt with future business needs.

                Data within the Database will be used to provide information to the users of the company the database is serving.  This could be in the form or reports that are used every day to provide information on things such as sales for the day, week or etc.  What are our Top Vendors, our Top Salesmen by Division and so forth.  Your database might be used to determine what products are not as profitable if it tracks that to determine what products you might want to discontinue or not.  When designing a database, and speaking with the owners of the information, these are the types of questions that you need to determine they the business users will need to ask of the database, and then the database can be designed to capture the characteristics that will provide the details/information to the users.

                What should be included in each table is a unique identifier that identifies each record in each table in most cases, which is the tables Primary key.  This Primary Key can be used in other tables to join related information together.  An example of this would be connecting employee information with their contact information.  An employee might have 2 or 3 or even more contact numbers, so you can have an Employee Table that houses employee name, and other general information such as employee number and so forth.  That employee number which should be the Primary Key could be used to link to the Contacts table that stores contact numbers as a Foreign Key.  This way if an employee hypothetically has 100 different contact numbers, they all could be related to the individual through the use of a Primary, Foreign Key relationship between the two tables.

                When designing a table you should never contain repeating or redundant information.  So using the example in the above paragraph, some people would say that you could just store 2 contact numbers and just store that information in the Employee table and not create a Contacts table.  And this scenario can work, but would be a limitation on your database.  So if you are a marketing company and reaching out to people is one of your most important goal, then you wouldn’t want to limit yourself to just two contact numbers, so separating out this information into more than 1 table would provide you with greater flexibility for your database.

                Within the Sakila database an example of a many to may relationship would be the Film Table to the Category table.  There could be many Films that contain many categories and many Caetories that are in many Films.  A many to many relationship cannot be expressed directly in a database design, so there needs to be a linking table between the two that breaks down this Many to Many relationship down to two 1 to Many relationships, which essentially expresses the Many to Many relationship using 3 tables.  Another example of this within the Sakila Database is between the Film and Actor tables.  Many films can have many actors and vice versa, so there is a Film\_Actor table between these tables that link them together.

                One problem about the World Database is that it contains information about many countries but only has 3 tables.  This in my opinion will always mean that this database in its current form will only be able to store a little bit of information that’s applicable to all countries.  Since each country can vary in the scope of information that can be related to it, it would be hard to create a database that a lot of different information about all countries on varying topics, that could apply to some but not all countries where the same type of data is required for all countries.  So when creating a database with this large of a scope for storing information, the scope of the database would have to be specific in nature and scope to be applicable to a large genre such as all the countries in the world.

Victoria Lasode



**Notes from Database Design Basics Article**

*(https://support.office.com/en-us/article/Database-design-basics-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5)*

A good database design ensures reduced redundancy in data to conserve space and reduces likelihood of errors and inconsistencies. It also maintains accuracy and integrity of data.

The following questions need to be addressed when creating a database:

What is the purpose of the database? How will it be used and for whom is it intended?  This provides a guide to the design and helps us to maintain focus throughout the design process.

It is equally important to know what information we will be presenting to our intended users. This will help to know the items we need to put in our database.

We should also break up every piece of information into useful bits.

A good practice is to always capture each fact once in a database. If the same information is being repeated in more than one place, such information should be stored in another table. Also results of calculations should not be stored in a table. Rather these calculations can be performed while querying the database and will be returned in the result. Another good practice is to break down information into logical parts to aid easy and quick retrieval of information. An example is to break down address into street, city, state and zip code.

Every table should have a field or set of fields that uniquely identifies each record stored in the table. This is called the primary key. A primary key cannot have duplicate values and cannot be null so a field that stores values that are likely to be duplicated cannot be chosen as the unique identifier. For example, in a table that stores people’s information, two or more people can share the same information such as last name, first name, date of birth etc. so such fields cannot be set up as the primary key.

In a database where there is more than one table, one table’s primary key can be used as a reference in other tables; it is known as foreign key in other tables. With this reference, any changes in the primary key must be changed in other tables where this key is referenced. This helps to keep the records in sync.

Creating table relationships is a way to bring information that has been divided into separate tables together. A type of table relationship is the **Many -to-many relationship**; for example, in the actor and film\_actor tables of the Sakila database, an actor can feature in many films, a film features many actors. Other relationship types include one- to- many, one-to-one.

Once the tables and relationships have been defined, we should create and insert values into our tables with sample data and test them out with queries to see if they produce intended results. This way, alterations that we need to make to the table design become obvious and we can go ahead to make the necessary changes.

Normalization can be applied after our design is completed to ensure that our information items have been divided into appropriate tables. We ensure that our design arrives at what is called a normal form. There are 5 widely acceptable normal forms, and these are first normal form (1NF), second normal form (2NF), third normal form(3NF), fourth normal form and fifth normal form but only the first three are required for most databases.

The following rules shows compliance for the normal form types:

In a 1NF, a row/column intersection can hold only a single value.

In a 2NF, each non-key column should be fully dependent on the entire primary (where it is composite, on all key columns), not just a part and,

In a 3NF, each non-key column is dependent only on the primary key.

Gary James

SD-102

2/26/2018

Database Design

As previously discussed in MongoDB about the difference and applications between NoSQL and relational databases, which provide functionality and scalability, depending on the nature of the company will help with designing a database. The uses for different database technologies depend on what the company is trying to accomplish and the intended use. The relational database work well with data warehouses, Web and mobile globe applications. For instance, if  a company has a large volume sets of data, semi-structured or unstructured, and service large number of user request, then NoSQL would be ideal. “Relational databases- RDBMS - is for enterprise OLTP and ACID compliance, or databases under 1 terabyte. NoSQL is for scaled OLTP and JSON documents. Hadoop is for Big Data Analytics (3).” However,  provided the company chooses the type of database that best suits their needs, will help determine what information will be needed to start building a database. Asking questions about the purpose of the database, the intended use, and who all will be using it, is the beginning for creating a database. By writing down the purpose and the necessary information needed will lay the groundwork for dividing and turning information into tables. So, what are tables? Tables are subjects that store facts or information about that particular subject into fields and rows. The information gathered about the subjects need to be sorted and broken down to make it easier to search and edit. The primary key(s) are unique identifiers for the each table column, which are added to an additional column that share the same column identifier, thus representing a one-to-many or one-to-one relationship because a one-to-one relationship table involve the use of a common column from another table. However, only one fact needs to be established for a particular table because if the same fact is used in multiple tables it would violate 1-NF and be considered redundant, which then would require another table, so the information doesn’t overlap. A foreign key joins two tables that have a many-to-many relationship to the parent table, which requires a third table, to break the many-to-many relationship into one-to-many. Once you have established the tables and columns, you need to ask yourself which tables share common column(s) that would represent a one-to-many relationship or many-to-many relationship and which would use the foreign key(s) to join the tables- foreign key(s) are another table’s primary key, usually the referring table. On the other side, there can be many-to-many relationships between tables, which require a “go between” table, that consist of foreign key(s), to join the two tables. For example, the Movies database, there are many-to-many relationships between the Movie and Cast, Movie and Character, and Movie and Genres tables. All of which would require a “go between” table because there can be a movie that has many cast members or many cast members for a movie. Same goes for Movie and Character and Genre tables. By creating the “go between” table, it breaks down the many-to-many relationship to a one-to-many by using the primary key from each table to add to the “go between” table, which would be considered as foreign keys in order to join the tables. Also, in the World database, there are only three tables- Country being the parent and the other two, City and Country Language, being the children- The Country and City table have a one to many relationship and do not require a “go between” table to break the relationships down, but the tables do have similar columns, which would use a foreign key, Country Code, in the City table to join to the parent table. As for the Country Language table, it has a one-to-one relationship and only requires a primary key to join to the parent table since they have a similar column, Country Code. However, the Country table has a lot of information stored in it and could be broken down into more tables to make a specific search and editing easier. After dividing and figuring out which tables to use, create a diagram to have a visual and edit the tables, taking out repeating columns. If you happen to see a stand-alone table, as seen in the Sakila database, Film\_text table, you would need to edit the table and establish a connection to the parent table. After reviewing and asking questions as previously discussed, you should be able to refine you database design and create a functional and fluid database with accurate information.

Works Cited

(2) Foote, Keith D.”A Review of Different Database Types: Relational versus Non-Relational”   
<http://www.dataversity.net/review-pros-cons-different-databases-relational-versus-non-relational/> (12/26/2017)  
  
(3) Okman, Lior & Gal-Oz, Nurit & Gonen, Yaron & Gudes, Ehud & Abramov, Jenny. (2011). Security Issues in NoSQL Databases. . 10.1109/TrustCom.2011.70.   
<https://www.researchgate.net/publication/254018091_Security_Issues_in_NoSQL_Databases>  (12/26/2017)