Database Designs

Section 5: Database Design Theory

5.1 Practical Tips on Database Design

A good database design is when a database can withstand the test of time. If the data structure does not change that much even when the data grows and expands into new scenarios/areas of information then this can signify a good good database design. This is a hard topic and so much can be said about this topic alone. Below are a few important principles of designing a good schema for a database.

Designing the tables of a database is called designing the schema of a database.The list below is not a comprehensive list but will help when setting up a database to withstand the test of time.

- Identify the problem you are trying to model/capture from the real worlds
- Have a good idea of the types of queries (i.e. information to drive business decisions) that you want to run on the database
- Break up the information into logical units where each logical unit is one table in your database
- Each logical component will probably connect with other logical components. Capture the relationships either via a constraint or other tables devoted to holding relationship specific data

Tables typically hold information about an entity or a relationship. Think of whether you need a way to uniquely identify each row in a table. If the answer to this question is yes (in most circumstances this will be true) you need a Primary Key.

- A Primary Key can never have a NULL value
- A Primary Key will be used to represent that row and other tables can reference that row by specifying the Primary Key value
- Do not use data that can change as a Primary Key value
- If there are no such values in the fields you want to store then you can generate a value which will be used as a unique identifier i.e. DBMS have auto-increment to generate unique identifiers for you
- Numeric values tend to be more efficient in representation and lookup and should be preferred for Primary Key values
- Having an identifier whose job is to be a key value is standard practice
- Remember, most DBMS will automatically create an index on the Primary Key for faster lookup and retrieval

Do columns have other constraints which they need to satisfy?

- Are NULL values allowed for any column? If the answer is no then the column should be specified as NOT NULL
- Should columns have a default value? If the answer is yes then the column should be setup with the DEFAULT keyword followed by what the default value for that column should be

- Ensure the column datatype are setup correctly e.g. don't use a string for storing date information (even though it is allowed) as this will limit the features applicable to date and time values for example comparing date difference between two date values
- Think about the number of characters/numbers a column will hold e.g. if a data is a single character don't use a CHAR(30) to store the string. Database occupy space and space that are not used up are a waste of resource and storage resource is expensive

5.2 Further Tips on Database Design (Table Relationships)

How many tables should we use to store information? To answer this question we should always think of the relationship between the different pieces of information being represented.

If the information is a one-to-one compulsory information e.g. students and their email address there is no need to store the information is separate tables. Every student has just one email address and therefore there is no need for the email address to be stored in a separate table from the rest of the students information.

When the relationship between the various pieces of information are one-to-one and the information is not optional (i.e. it is always specified) then we can put that information into the same table. Remember this as a rule of thumb. You may want to separate it for some reason but typically if the information has a one-to-one relationship there is no reason to separate the data.

StudentID	Name	Email
1	John Doe	j.doe@email.com
2	Martha Jones	m.jones@email.com

If the information is a one-to-one but one of the information is optional e.g. students and siblings information you may decide to hold these information in separate tables. A student may or may not have siblings and if the main students table included sibling information then there would be many empty cells. In this case the students table will hold only the mandatory information while the sibling information should be held in a separate table.

When the relationship between the various pieces of information are one-to-one but some information are optional (i.e. may or may not exist) then it may be best to separate the information keeping all the mandatory information in one table and the optional information in another. The main table will hold a Primary Key as an identifier for each row. The second optional data table would reference the main table's Primary Key value as a Foreign Key value, thereby linking both the table information together.

Optional data are represented by the lack of rows in the second table and not by empty cell value in a row. This is a much cleaner setup.

StudentID	Name	Email
1	John Doe	j.doe@email.com
2	Martha Jones	m.jones@email.com

StudentID	Sibling Name
1	Joanne Doe
1	Jonathon Doe
2	Katey Jones

If the information is a one-to-many or a many-to-one relationship e.g. employees and managers information then at a minimum we would need two tables to represent this information. One table would be used to store all information relating to employees and another to hold the relationship between an employee and his/her manager.

Notice in the example table below, the second table uses only the unique identifiers to specify the relationship. Once again, the Primary Key is used on the main table to represent the entire row details of the employee. The second table merely makes reference to these keys only.

This is not the only way to represent this information and there are different ways of representing this data but typically speaking a one-to-many or many-to-one relationship is best represented using two tables at a minimum and what the two tables contain can be designed differently.

EmployeeID	Name	Email
1	John Doe	j.doe@email.com
2	Martha Jones	m.jones@email.com
3	Katey Jones	k.jones@email.com

EmployeeID	Manager_EmployeeID
1	2
3	2

Finally if the information is a many-to-many relationship e.g. stores, products and revenue information then we would need 3 or more tables to represent the information. The revenue data is specified for every product sold in every store and therefore links the stores and products data in a many-to-many relationship.

In such a case we would need at a minimum 3 tables, the first table representing the first entity's information, the second table representing the second entity's informations and the third table linking the two entities tables in a many-to-many relationship.

StoreID	StoreLocation
1	Birmingham
2	Coventry

ProductID	ProductName
1	Bread
2	Milk
3	Nutella

StoreID	ProductID	Date	Revenue
1	1	30 November 2020	453.22
1	3	30 November 2020	225.55
2	2	30 November 2020	1860.98

It is best to keep the entities information in separate tables. If we were to add more details about the product e.g. product details, if we did not have a separate table for stores and products, we would end up adding more and more information to the single table making the table fatter even if the information has nothing to do with the store itself. Separate tables allows us to expand the information logically while still maintain tall and thin structures. The linking table itself will have a tall and fat structure.

The linking table would use Foreign Keys to link the the Primary Key values of the other entity tables which combines all three tables together in a many-to-many relationship. This provides a clean structure for specifying a many-to-many relationship.

Understanding the relationships of the information you are trying to represent is vital in setting up the tables and its column in the best way possible to stand the test of time.