

Creating a Software Solution



Designing a Database for Maintenance and Scalability

Our Story so Far

- **We Found A Problem To Solve**
- **We Built Requirements and Use Cases**
- **We Created A Clickable Demo**
- **We Created a User Experience**

Next Step

It is time to start implementing a solution

**Sometimes we start from the Front End
sometimes from the Back**

**In this case, our focus is the data we want to
capture and report on**

The Importance of The Data

- **This application is all about goal tracking and reporting**
- **It is critical to the usefulness of the project to properly store and retrieve data**
- **This is a great next step from a first clickable demo**
- **Data comes from use cases, not interfaces**

Database Overview

A database has a few key concepts and goals:

- **Store Data**
- **Retrieve Data**
- **Relate Data**

The Structure

Databases use different terminology

There are always (almost) the concept of a table, columns, and a row

There are primary keys, foreign keys, indices, and general constraints/triggers

These structures/features provide integrity

Columns

A Table consists of rows and columns

A column has a type, size, name, and possibly a default value

Columns provide storage and no other rules except potentially a default value

Rows

They are defined by the columns

Data is stored in a row

**Columns provide structure and then data is
stored row by row**

Designing A DB

- **Consider the Core Data Structures (Tables)**
- **Consider reference/lookup data**
- **Relationships among data**
- **1 to 1, 1 to many, many to many (xref tables)**
- **Support tables and Reporting**

Data Duplication and Normalization

- **Normalized data removes duplicate values**
- **Some normalization is recommended for all non reporting tables**
- **Normalization increases scale and flexibility, but reduces time to lookup and insert**

Building Your Core

- **Determine your main actors/groups of data**
- **This is often a User, Account info, Products, Services, Customers, and similar data**
- **Look to your use cases for ideas and make sure they are supported**

Lookups and References

- **Lookups are often simple structures for a list of values (LOV)**
- **Examples include states, countries, status values, customer types, etc.**
- **Sometimes more than an id and a value are needed, consider sort order, active/inactive, and short vs long names or labels**

Relationships

- **Look for use case relationships like “is a”, “has a”, and “uses a”**
- **Normalizing data will create relationships**
- **Generally you want to avoid one to one tables unless it also includes a one to zero possibility (is a)**

Relationships

- **One To Many relationships are common and include lookups as well as sets of data (phone numbers, accounts, orders, etc.)**
- **Many To Many is typically used for multiple views of data (groups of objects where an object can be in multiple groups). This requires a cross reference table.**

Support/Report

- **Structures can become complex so a simplified copy can be very useful**
- **These are often denormalized for speed of recovery**
- **There is an overhead cost of keeping data in sync across multiple sources**
- **Use these sparingly or in another DB (warehouse)**

Best Practices

- **Avoid meaningful primary keys**
- **Name tables and columns in a human readable way**
- **Keep sizes as small as possible/reasonable**
- **Avoid magic values**

Best Practices

- **Avoid columns that have more than one meaning**
- **Include foreign keys when possible**
- **Avoid circular references**
- **Simplify/Normalize where it can save space, but avoid forcing too many joins to retrieve data**

Bottom Line

- **Start From Major Objects/Actors**
- **Add Details as needed (properties, configuration, etc.)**
- **Review for relationships and normalization that can simplify maintenance and reduce size**
- **Follow Best Practices**

Thanks!

Send any questions, comments, or requests for assistance to info@developpreneur.com or contact us on the site. We are available to help you build your solution at any point in the process.