Brett Fuller

10/27/2024

CSD 310 – Assignment 1\_3

In a database a relationship is how two tables in a database associate with one another. One of the most common forms of database relationships is the one to one relationship where one item is related to one other item. Take for example a social security number. In this instance an American has only one social security number and that social security number only belongs to one person. In this example there might be a table of people and a table of social security numbers. In the table of people there would be a field that contains the foreign key of the social security number in that table. From there it is a very simple lookup to grab the social security number by directly referencing that entry in the SSN table instead of having to do a more complex lookup. This reference does not have to exist on both sides but it can and often does, which means that in the SSN table there would be an entry for the person’s foreign key allowing a reciprocal direct lookup. Another common type of relationship is a one to many relationship. In this type of relationship you will find an entity that has a direct association with multiple examples in another table. An example of this might be a teacher to classes. In this instance there would be a table of teachers and a table of classes. Each class would have a link to a single teacher, but each teacher might teach several classes. So then each class only links back to one teacher and each teacher would link back to multiple classes. The other most common form of relationship would be a many to many relationship, where the entities in a table would have multiple associations with entities in another table, and those entities would have associations with multiple entities in the initial table. A good example of this would be a table of students and a table of classes. In this example a student is likely enrolled in multiple classes and so that students would have references to each of the classes that they are enrolled in. Whereas each class would likely have multiple students enrolled in them and so they would have references to multiple students.

There are several advantages to relational databases; one of the most obvious examples is being able to maintain performance while having high levels of organization. By using relationships between tables you are able to divide your data into smaller individual sets of entities and having direct references between the tables. Without these relationships it can require more intensive lookups between tables that may require both more code and may take more compute intensive searches to uncover the nonspecifically defined relationships.  These direct relationships also help to eliminate redundancy in the data. If you do not have direct relationships it may be tempting to try and maintain a set of data in multiple locations in order to make it easily referenced from multiple areas. There are also several disadvantages to relational databases. Relational databases require a lot of structure which can require a lot of upfront planning and can be more difficult to maintain as the dataset grows larger.  It can also be extremely difficult to maintain when dealing with large amounts of unstructured data. Furthermore as the database outgrows a single server and starts having to be maintained along large clusters of servers that may exist over large geographic areas the performance can degrade and make it impractical for these types of environments.

NoSQL provides several advantages and disadvantages that are almost antithetical to relational databases which are also of an SQL variety which I guess is why it is so appropriately named. NoSQL is specifically designed to work with any type of dataset, being it structured, unstructured, or semi structured. It has easy to maintain and update schemas if it contains any schema at all and allows for massive scalability. However, not everything is rosy with a NoSQL database, there is a lack of standardization around NoSQL databases, each of the different forms have very different schemas or none at all can make it an intimidating landscape. There is also a lack of consistency; the data is not necessarily unique in a database leading to the possibilities of invalid data. Additionally the lack of joins makes queries more difficult and can consume more compute.

MySQL has many features that make it appealing for developers. One of the largest ones is that it is platform independent. This means that it can be ran on a variety of different operating systems from windows, to linux, to UNIX, BSD, to macOS. It also means that it can interface with a large variety of programming languages making it an incredibly versatile database variety. MySQL is Free and open source so it is a good pick for many use cases though there are some restrictions for use as defined by the GPL. It is highly performant making it a good choice for applications that need a high performance database.

MongoDB is another popular database type that offers a wide variety of appealing features. MongoDB is built upon the Document Model wherein data is stored as documents and are organized into collections. This allows the documents to be treated as objects which is a familiar concept for most developers. Mongo is incredibly scalable because of the concept of sharding, where data is split up into smaller instances or shards that can then be distributed across a large number of servers, allowing for horizontal scaling. From here Mongo makes heavy use of replication allowing for shards to be maintained over multiple instances across multiple servers, allowing for redundancy in that losing a server does not put data at risk while also allowing for data locality and having instances of the data maintained close to where the data might be accessed.

**References**

Foote, K. D. (2022, November 17). *NoSQL Databases: Advantages and Disadvantages - DATAVERSITY*. Dataversity. Retrieved October 27, 2024, from https://www.dataversity.net/nosql-databases-advantages-and-disadvantages/

Gillis, A. S. (n.d.). *What is a relational database? | Definition from TechTarget*. TechTarget. Retrieved October 27, 2024, from https://www.techtarget.com/searchdatamanagement/definition/relational-database

MongoDB. (n.d.). *MongoDB Features & Key Characteristics | MongoDB*. MongoDB. Retrieved October 27, 2024, from https://www.mongodb.com/resources/products/capabilities/features

Tutorials Point. (n.d.). *MySQL - Features*. TutorialsPoint. Retrieved October 27, 2024, from https://www.tutorialspoint.com/mysql/mysql-features.htm

Zivkovic, M. (2024, February 12). *Database Relationships: The Ultimate Guide for Beginners*. Luzmo. Retrieved October 27, 2024, from https://www.luzmo.com/blog/database-relationships