4 What is a Relational Database?

Database Management Systems

Databases are more then just stores of data. They are highly organized files that allow for data to be input, organized, and retrieved efficiently. The data is placed into tables where it can be sorted and filtered in flexible ways.

Database management systems can not only search and manipulate tables of data but also help create and organize them so they are super efficient and work properly so the right data can be retrieved to become useful information.

With Relational Database Management Systems, we can create, and manipulate our database as well as control the hardware that stores the data with these systems. A RDBMS can let you:

- Create the Structure and the Rules for the data
- Store, Manipulate, and Sort Data
- Create users with different levels of access to the data or giving them roles
- Backup and secure the data

Relational Databases

In relational databases, data is stored in one or more tables. Each table represents a real-world entity or one group of related items. These tables are made of rows and columns. Each column header describes what fields of data will be in the table or the attributes of

the entity. So each column has fields and each row is a record or one instance of that entity.

Most tables have a column that uniquely identifies each row in the table. You can define this as the primary key. Each primary key must be unique and never repeated in the table where it is defined as the primary key.

Some tables will combine two columns that together make up the primary key. These are called composite keys.

What makes a relational database, is the relationship between tables. A primary key can relate to a foreign key of another table. These relationships are enforced with referential integrity. This means that any changes to the data in the database won't create invalid relationships between the tables.

- Primary Key - uniquely identifies each row in a table. It can't be repeated as a primary key.
- Foreign Key - show a relationship to a primary key of another table. It could possibly be repeated as a foreign key depending on the relationship.
- Composite Key - when two or more columns are used to uniquely identify a row in a table
- Natural Key - uniquely identifies a single record or row in a table and is made up of real data (data that has meaning and occurs naturally in the real world)
- Surrogate Key - uniquely identify a single record or row in a table but it doesn't have a natural relationship with the rest of the columns in the table. They can be created with auto_incrementing.

Other keys can can be used when defining columns in a table. In Workbench they are seen as the PK (Primary Key), NN (Not Null), UQ (Unique), BIN (Binary), UN (Unsigned), ZF (Zero Fill) and AI (Auto Increment). In this course we will understand and use PK, NN and AI. The UN or Unsigned is a good option to make sure surrogate keys have a wide range of positive numbers, but is not necessarily needed in this course due to our relatively small databases, but would be a good idea to use on larger databases. It extents the range of possible positive key values that can be used.

For more information about these keys you can visit: MySQL Workbench Data Modeling & Development Chapter 4 pages 140-143

Entity Relationships

Sometimes the term cardinality is used to describe the different types of relationships that can occur between tables. Cardinality refers to how the occurrences in one table are linked to the number occurrences in another table. There are 3 types of relationships. One-to-one, one-to-many, and many-to-many.

A one-to-one relationship is not as common as the other two relationships. Consider an employee table that everyone in the company has access to but there are some attributes of the employee you want kept private like their salary, social security number, or home address. For security and privacy reasons you might want that information stored in a separate table of the database. There would be a one-to-one relationship here because each employee would have one and only one private information about them. And the private information about each employee belongs to one and only one employee.

A one-to-many relationship is more common. An example of a one-to-many might be two entities such as the departments of a company and the employees who work for that company. Each department has employees and each employee belongs to a department within that company. We would be assuming here that each employee works for only one department. The primary key of each department is related to the foreign key of each employee. A foreign key value can be repeated many times in the table for each employee that belongs to that department.

A many-to-many relationship is also common. An example of a many-to-many might be between two entities such as classes and students. Each class can have one or more students in the class and

each student can take one or more classes. Relational database systems don't allow you to implement a direct many-to-many relationship. A many-to-many relationship always has to be resolved or broken down into 2 one-to-many relationships. This provides a much better data structure for the database. Another table is created in the process. This table that is created is referred to as a linking table, or a joining or a bridge table. For example the table between students and classes would be something like an enrollment table showing which student is enrolled in which class.
