Relational Databases

There are a number of different types of models of databases but the relational model is by far the most common and is the standard for database applications. This type of database reduces data redundancy and make data retrieval efficient.

In relational databases data is stored in one or more tables. Each table will represent a real-world entity or one group of related items. These tables are made of rows and columns. Each column header will describe what fields of data will be in the table or the attributes of the of the entity. This one will represent college classes. Later when you are using a entity-relationship (ER) diagram the table will be represented with the column headers listed in the entity like this. Each row's worth of data represents one record of that entity. In our classes table it represents one class per row.

So each column has fields and each row is a record or one instance of that entity. Where the columns and rows intersect is the cell where one single data entry is stored. Here it is the data storing the value of room 335.

Most tables have a column that uniquely identifies each row in the table you can define this as the primary key. The primary key must be unique and never repeated in the table where it is defined as the primary key. Here the class_id is unique for each row in the table. It would be the primary key. In the Entity relationship diagram, the primary key will usually have a small key icon next to it.

Some tables will combine two columns that together make up the primary key. These are called composite keys. Other keys include non-primary keys or unique keys that, like primary key, will also uniquely identify each row in the tables. You can also set indexes on columns that will provide an efficient way to access data from the table based on that column.

Here is an example from the text showing the vendors table from the ap database. Each row contains a vendor. The primary key is the vendor_id.

What makes it a relational database is the relationship it has between tables. Here is the same vendor table but this is showing another table, the invoices tables, and how it is related to the vendor table. For each vendor there can be multiple invoices. This is called a one-to-many relationship. There are also one-to-one relationships but they are not common and many-to-many relationships that are broken down to one-to-many relationships. We will visit those different relationships again later.

Notice how the vendor primary key, which is vendor_id, is used to relate the two tables but is repeated multiple times in the invoices table. (one-to-many) The invoice table has its own primary key that is not repeated but when a primary key is used to relate to another table then it is the foreign key of that other table. So the

primary key of the vendor table is now the foreign key of the invoices table. It can be repeated as many times as needed. In this case every time there is an invoice created for Federal Express Corporation the Federal Express Corporation's primary key, 123, is used as a foreign key of those invoices and that creates a relationship between the vendors and the invoices tables. So we can easily see the relationship of all of Federal Express Corporations multiple invoices.

When you define the foreign key for a table you have that foreign key enforce referential integrity. This means that any changes to the data in the database won't create invalid relationships between the tables. In this case referential integrity wouldn't allow us to add an invoice for a vendor that doesn't already exist. Also you can't delete a vendor if they have invoices that are related to them. You'd have to delete the invoices first before you could delete the vendor. This is referential integrity.

This is a screen shot showing the columns of the invoice table. This is showing the design view of that table and what was defined as the table was created. In this view you see each column name and the Datatype that goes with that field or attribute that belongs to that column. The invoice_id for example has a data type of Integer with length of 11. You can also see the checkboxes for Primary key, and not null (not null means that you cannot leave this field blank, it is required. So as someone is entering a row or record of information into that table they cannot leave the invoice_id blank. There are other data types as well like variable character, date, decimal etc. We will study these in depth in another chapter but realize that there can be many different types of data whether that's numbers, text, dates, etc. Also notice that Payment_total and Credit_total have 0.00 as a default value This would be a value that would be used automatically if no other value is provided when the a row is added to the table. The checkbox AI stands for Auto-Increment. An auto increment column is a numeric column whose value is generated automatically by the system as the row is added to the table. Also called a surrogate key.

This is an EER (Enhanced Entity Relationship) Diagram. It shows how the tables are defined and related. This shows the ap database that contains 5 related tables. The two tables we were looking at earlier were the vendors and invoices tables. Each of the columns names for the tables are listed. It also shows the datatypes and a symbol next to each column name that identifies it. The key shows that it is a primary key field and the red diamonds show that it is a foreign key field. There are connector symbols between the tables as well. The symbol closest to the invoices table indicates that many invoices can exist for each vendor. The connector symbol next to the vendors tables shows that only one vendor can exist for each invoice.

Notice the invoice_line_items table and how it uses two different keys to uniquely identify the rows in that tables. This is an example of a composite key.

Let's review the different types of keys we learned. The Primary key uniquely identifies each row in a table. It can't be repeated as a primary key. A 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. A composite key is when two or more columns are used to uniquely identify a row in a table. A 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) Social security numbers for people, or ISBN numbers for books are good examples of natural keys. They occur naturally but are always unique for each person or book.

Surrogate keys also uniquely identify a single record or row in a table but they don't have a natural relationship with the rest of the columns in the table. The surrogate key is like the auto-increment that we talked about earlier, it is a value generated automatically as the record is inserted into a table. They are commonly a numeric number. So which is better? There are pros and cons to both choices and neither is right or wrong.

So there we have an introduction to relational databases and some of the vocabulary that you will hear as you work with them.