**Relational Database Concepts**

In this video, we will learn about different types of models, how we use models to map data to tables, and define relationships between tables. At the end of this lesson, you will be able to explain the advantage of the relational model, explain how the entity name and attributes map to a relational database table, describe the difference between an entity and an attribute, identify some commonly used data types, and describe the function of primary keys. The relational model is the most used data model for databases because this model allows for data independence. Data is stored in a simple data structure. Tables: this provides logical data independence, physical data independence, and physical storage independence. An entity relationship data model, or ER data model, is an alternative to a relational data model. Using a simplified library database as an example, this figure shows an entity relationship diagram or ERD that represents entities called tables and their relationships. In the library example, we have books. A book can be written by one or many authors. The library can have one or many copies of a book. Each copy can be borrowed by only one borrower at a time. An entity relationship model proposes thinking of a database as a collection of entities rather than being used as a model on its own. The ER model is used as a tool to design relational databases. In the ER model, entities are objects that exist independently of any other entities in the database. The building blocks of an ER diagram are entities and attributes. An entity can be a noun: person, place, or thing. In an ER diagram, an entity is drawn as a rectangle. Entities have attributes which are the data elements that characterize the entity. Attributes tell us more about the entity. In an ER diagram, attributes are drawn as ovals. Using a simplified library as an example, the book is an example of an entity. Attributes are certain properties or characteristics of an entity and tell us more about the entity. The entity book has attributes such as book title, the edition of the book, the year the book was written, etc Attributes are connected to exactly one entity. The entity book becomes a table in the database and the attributes become the columns in a table. A table is a combination of rows and columns. While mapping, the entity becomes the table. Having said that, the table has not yet taken the form of rows and columns. The attributes get translated into columns in a table providing the actual table form of rows and columns. Later, we add some data values to each of the columns, which completes the table form. Each attribute stores data values of different formats, characters, numbers dates, currency, and many more besides. In the book table example, the title is made up of characters. As book titles vary in length, we can set the variable character data type for the title column: VAR char. For character columns that do not vary in length, we use character or char. The Edition and year columns would be numeric. The ISBN column would be carved because it contains dashes as well as numbers and so on. Using the book entity mapping as an example, we can create the tables for the remainder of our simplified library example using entity names, like author, author list, borrower, loan, and copy. The entity attributes will be the columns of the tables. Each table is assigned a primary key. The primary key of a relational table uniquely identifies each tuple or row in a table, preventing duplication of data and providing a way of defining relationships between tables. Tables can also contain foreign keys which are primary keys defined in other tables, creating a link between the tables. Now you know that the key advantage of the relational model is logical and physical data independence and storage independence. Entities are independent objects which can have multiple characteristics called attributes. When mapping to a relational database, entities are represented as tables and attributes map to columns. Common data types include characters such as car and VAR char, numbers such as integer and decimal, and timestamps including date and time. A primary key uniquely identifies a specific row in a table and prevents duplication of data.

**Types of SQL statements (DDL vs. DML)**

At the end of the video, you will be able to distinguish between data definition language statements and data manipulation language statements. SQL Statements are used for interacting with Entities (that is, tables), Attributes (that is, columns) and their tuples (or rows with data values) in relational databases. SQL statements fall into two different categories: Data Definition Language statements and Data Manipulation Language statements. Data Definition Language (or DDL) statements are used to define, change, or drop database objects such as tables. Common DDL statement types include CREATE, ALTER, TRUNCATE, and DROP. CREATE: which is used for creating tables and defining its columns; ALTER: is used for altering tables including adding and dropping columns and modifying their datatypes; TRUNCATE: is used for deleting data in a table but not the table itself; DROP: is used for deleting tables. Data Manipulation Language (or DML) statements are used to read and modify data in tables. These are also sometimes referred to as CRUD operations, that is, Create, Read, Update and Delete rows in a table. Common DML statement types include INSERT, SELECT, UPDATE, and DELETE. INSERT: is used for inserting a row or several rows of data into a table; SELECT: reads or selects row or rows from a table; UPDATE: edits row or rows in a table; And DELETE: removes a row or rows of data from a table. Now you know that: DDL or Data Definition Language statements are used for defining or changing objects in a database such as tables. And DML or Data Manipulation Language statements are used for manipulating or working with data in tables.

**CREATE TABLE Statement**

At the end of this video, you will be able to explain how the entity name and attributes are used to create a relational database table. Now let's look at the most common DDL statement: Create. The syntax of the CREATE table is shown here: You start with “CREATE TABLE” followed by the name of the table you want to create. Then enclose rest of the statement inside a pair of parenthesis or round brackets. Each row inside the parenthesis specifies the name of a column followed by its datatype and possibly some additional optional values that we will see later. Each attribute or column definition is separated by a comma.

For example, if we want to create a table for provinces in Canada you would specify: CREATE TABLE provinces (id char(2) Primary key not null comma name varchar(24) close parenthesis. In this example, the data types used are: CHAR which is a character string of a fixed length, in this case 2. And VARCHAR, which is a character string of a variable length. In this case, this variable character field can be up to 24 characters long. Issuing this statement would create a table in the database with 2 columns. The first column id for storing the abbreviated 2 letter province short codes such as AB , BC, etc. And the second column called name for storing the full name of the province, such as ALBERTA, BRITISH COLUMBIA, etc. Now, let’s look at a more elaborate example based on the Library database. This database includes several entities such as AUTHOR, BOOK, BORROWER, etc. Let’s start by creating the table for the AUTHOR entity. The name of the table will be AUTHOR, and its attributes such as AUTHOR\_ID, FIRSTNAME, LASTNAME, etc. will be the columns of the table. In this table, we will also assign the Author\_ID attribute as the Primary Key, so that no duplicate values can exist. Recall, the Primary Key of a relational table uniquely identifies each tuple (or row) in a table.

To create the Author table, issue the following command: CREATE TABLE author ( author\_id CHAR(2) PRIMARY KEY NOT NULL, lastname VARCHAR(15) NOT NULL, firstname VARCHAR(15) NOT NULL, email VARCHAR(40), city VARCHAR(15), country CHAR(2) ).

Note that the Author\_ID is the Primary Key. This constraint prevents duplicate values in the table. Also note that Last Name and First Name have the constraint NOT NULL. This ensures that these fields cannot contain a NULL value, since an author must have a name.

Now you know that: CREATE is a DDL statement for creating Entities or tables in a database. The CREATE TABLE statement includes definition of attributes of columns in the table, including Names of columns; Datatypes of columns; and other Optional values if required such as the Primary Key constraint.

**ALTER, DROP, and Truncate tables**

After watching this video, you will be able to: Describe the ALTER TABLE, DROP TABLE, and TRUNCATE statements. Explain the syntax. Use the statements in queries. You use the ALTER TABLE statement to add or remove columns from a table, to modify the data type of columns, to add or remove keys, and to add or remove constraints. The syntax of the ALTER TABLE statement is shown here. You start with ALTER TABLE followed by the name of the table that you want to alter. Differently to the CREATE TABLE statement though, you do not use parentheses to enclose the parameters for the ALTER TABLE statement. Each row in the ALTER TABLE statement specifies one change that you want to make to the table. For example, to add a telephone number column to the AUTHOR table in the Library database to store the author’s telephone number, use the following statement: ALTER TABLE author ADD COLUMN telephone\_number BIGINT; In this example, the data type for the column is BIGINT which can hold a number up to 19 digits long. You also use the ALTER TABLE statement to modify the data type of a column. To do this, use the ALTER COLUMN clause specifying the new data type for the column. For example, using a numeric data type for telephone number means that you cannot include parentheses, plus signs, or dashes as part of the number. You can change the column to use the CHAR data type to overcome this. This code shows how to alter the author table: ALTER TABLE author ALTER COLUMN telephone\_number SET DATA TYPE CHAR(20 ); Altering the data type of a column containing existing data can cause problems though if the existing data is not compatible with the new data type. For example, changing a column from the CHAR data type to a numeric data type will not work if the column already contains non-numeric data. If you try to do this, you will see an error message in the notification log and the statement will not run. If your spec changes and you no longer need this extra column, you can again use the ALTER TABLE statement, this time with the DROP COLUMN clause, to remove the column as shown: ALTER TABLE author DROP COLUMN telephone\_number; Similar to using DROP COLUMN to delete a column from a table, you use the DROP TABLE statement to delete a table from a database. If you delete a table that contains data, by default the data will be deleted alongside the table. The syntax for the DROP TABLE statement is: DROP TABLE table\_name ; So, you use this statement: DROP TABLE author; to remove the table from the database. Sometimes you might want to just delete the data in a table rather than deleting the table itself. While you can use the DELETE statement without a WHERE clause to do this, it is generally quicker and more efficient to truncate the table instead. You use the TRUNCATE TABLE statement to delete all of the rows in a table. The syntax of the statement is: TRUNCATE TABLE table\_name IMMEDIATE; The IMMEDIATE specifies to process the statement immediately and that it cannot be undone. So, to truncate the author table, you use this statement: TRUNCATE TABLE author IMMEDIATE; In this video, you learned that: The ALTER TABLE statement changes the structure of an existing table, for example, to add, modify, or drop columns. The DROP TABLE statement deletes an existing table. The TRUNCATE TABLE statement deletes all rows of data in a table.