CIT 111

Prepare 02: Data Storage and Relational Databases

This week you will...

- Study different types of databases, how data is stored, and what datatype the data belongs to.
- Examine Entity-Relationship Diagrams (ERD).
- Learn about Structured Query Language (SQL).
- Learn basic Database design principles.
- Use MySQL Workbench to create an Entity Relationship Diagram (ERD).
- Begin to understand how entities in our database are related to other entities and what attributes belong to each entity.

Step 1: Terms to Know

Familiarize yourself with the following terms.

Datatype:
A particular kind of data item, as defined by the values it can take.

Data Model:
Determines the logical structure of a database and determines in which manner data can be stored, organized, and manipulated.

Hierarchical Model:
Organizes data in a tree-like structure with parents and child nodes. Each child node will only have one parent node. Parents can have many child nodes.

Network Model:
An extension of the hierarchical database model. With the network model, child nodes can have more than one parent, allowing more flexibility.

Relational Model:
The most common model that organizes data in rows and tables, which in turn can be accessed and linked to other rows and tables by sharing a common field.

Object-Oriented Model:
Represents data as an object with properties and methods and combines database capabilities with object-oriented programming language capabilities.

NoSQL:

An unstructured database model that can store virtually any structure where new columns can be created easily.

Edgar F. Codd:

A computer scientist who invented the relational model for database management, the theoretical basis for relational databases and relational database management systems.

SQL:

Structured Query Language, the language to communicate with a database.

MySQL:

The most popular open-source relational database management system.

MySQL Workbench:

A visual tool for database providing data modeling, SQL development, administration tools, user administration, and backups.

Table:

A part of a database that represents one real world entity or one group of related instances of that entity. Each row of a table is one instance of the entity. Each column an attribute of that entity.

Primary Key:

An attribute (or group of attributes) that uniquely identifies a given entity in a table.

Foreign Key:

A primary key of one table that appears as an attribute in another table and acts to provide a logical relationship between the two tables. Foreign keys can repeat.

Referential Integrity:

A rule that prevents invalid relationships between tables when changes are made to the data.

Null:

A term stating that an attribute value doesn't have to exist. In other words, the user can leave the value blank.

Auto-increment:

A table column which produces a sequential system generated number when a new record is inserted into a table.

Surrogate Key:

A system-assigned primary key, generally numeric and auto-incremented.

Composite Key:

Two or more attributes that together define the primary key by unique combinations of their values.

Natural Key:

A key that naturally occurs in the attributes of an entity, such as a student ID or a course name.

ERD (Entity Relationship Diagram):

A diagram that shows how database tables are defined and relate.

Cardinality (Entity Relationships):

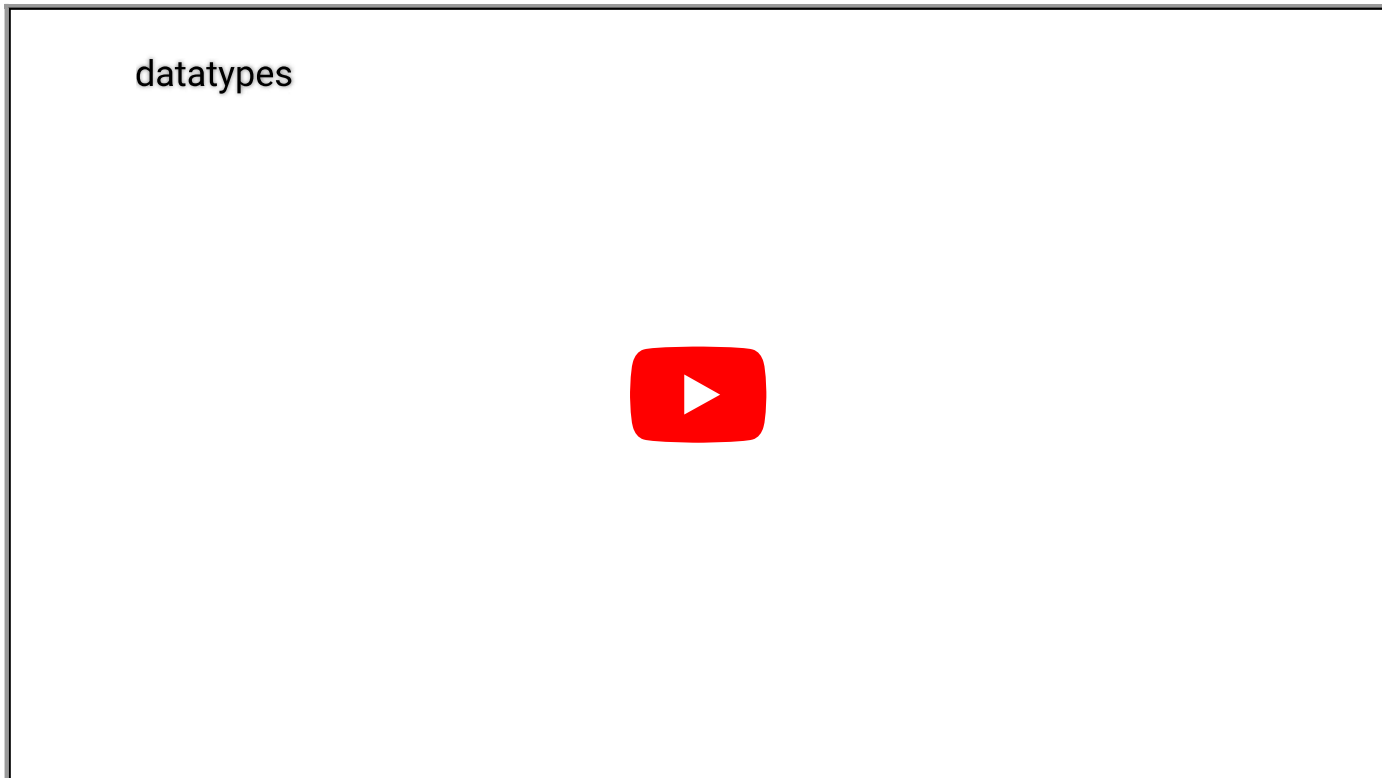
How the occurrences in one entity are linked to the number of occurrences in another entity (one-to-one, one-to-many, and many-to-many).

Crows feet notation:

A way to symbolize relationships between tables in an entity relationships diagram (ERD).

Step 2: Datatypes

Watch Data Types video:



(16:18 mins, "Datatypes" Transcript)

Step 3: Data Storage

How is Data is Stored?

Businesses depend on data stored in databases. For example, consider Amazon.com. It has a vast amount of data. When you search for a product, your search text becomes part of a question that is asked of the data (a query for their database), and the results are sorted and filtered down to what you are looking for. The Church of Jesus Christ has huge stores of data that it uses for family history or for church organizations and membership records. Social media sites store vast amounts of information on its many users. The financial world relies on billions of transactions that are stored in databases. Banks, companies, clients, customers, vendors, orders from all sorts of retail transactions all rely on databases.

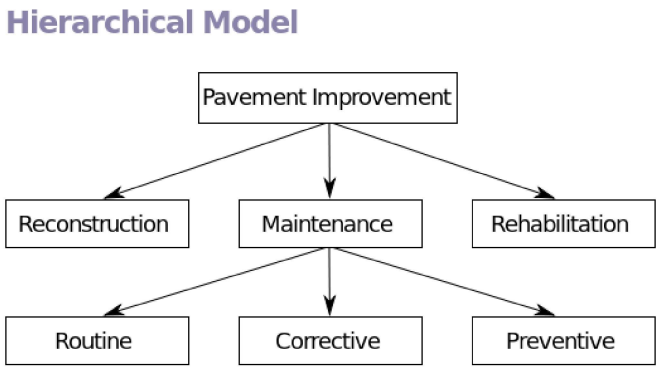
Companies can also make decisions regarding their next moves by querying their data, seeing trends, patterns, and then plan accordingly. We are surrounded by data, and it is growing every day. There is a need for people who can organize this data in a useful way so it can be used productively.

Database Models

A database model determines the logical structure of a database and determines in which manner data can be stored, organized, and manipulated. There are different types of database models for structuring a database.

Older models: network and hierarchical

A hierarchical database organizes data in a tree-like structure with parent and child nodes. Each child node will have only one parent node. Each parent node can have many child nodes. When data needs to be retrieved the whole tree is searched. This model was widely used during the mainframe era in the 1960s. It can still be used for storing file systems and geographic information. It allows for high performance needs, but its structure is very rigid.

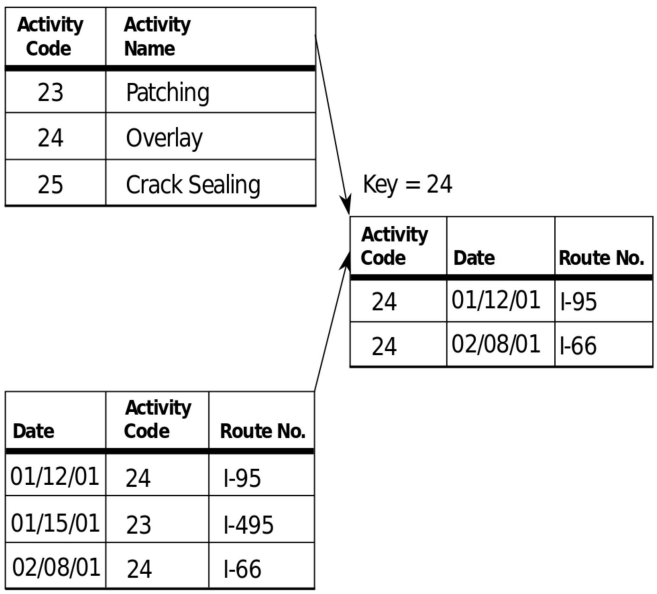


The network model is an extension of the hierarchical database model. With the network model, each child node can have more than one parent. This allows more flexibility. The network model was widely used before the relational model was introduced. Unfortunately, it was difficult to implement and maintain.

Relational Model

The relational database model was introduced in the 1970s. A relational database model organizes data in rows and tables, which in turn can be accessed and linked to other rows and tables by sharing a common field. The relational model is by far the most common type of database model, and that is what we will be studying for the rest of this course, but there are other models to be aware of and understand.

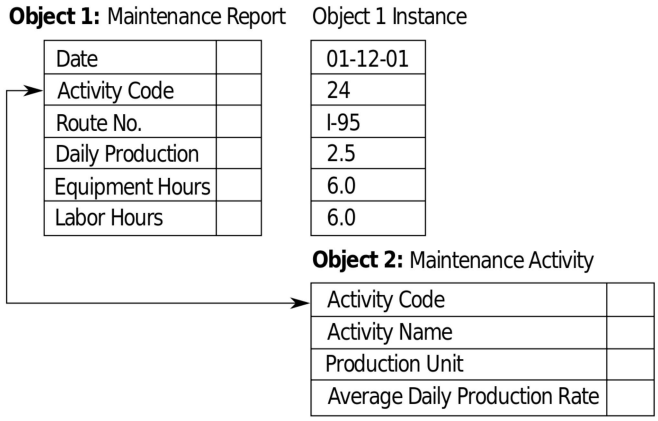
Relational Model



Object-oriented models

The term object-oriented database model was first introduced in the mid 1980s. An object-oriented database (OODB) is a database that represents data as an object with properties and methods. The object-oriented database model is an alternative to the relational model and uses a combination of object-oriented and relational database principles to process data. This model combines database capabilities with object-oriented programming language capabilities.

Object-Oriented Model



Semi-structured or Unstructured data models

With the emergence of 'Big Data' in the late 2000s, the unstructured database had its beginning. Some databases have such a large scale that a relational database model begins to have limitations. A very large-scale relational database has to be distributed on multiple servers. Handling tables across different servers

can be difficult. Also, the type of data you need to store might not fit into structured relational tables easily or might require a very complex structure to make it work. In these cases, a semi-structured or unstructured database model might be used. NoSQL is a type of unstructured database model. NoSQL is an abbreviation for 'Not Only SQL'. A NoSQL database can store virtually any structure. New columns can be created easily. In fact, a NoSQL database doesn't even need a predefined schema (or data model).

Relational databases just weren't designed to handle the quantity of data, number of users, and ever-changing data requirements for large systems like Amazon and Facebook. Even the church has used NoSQL for years to process billions of transactions. The amount of data volume in these types of systems just can't be managed by a single relational database system. NoSQL databases can provide high performance at an extremely large scale. They can handle billions of queries in a small amount of time. NoSQL databases also have far more relaxed data model restrictions.

There are, however, limitations to NoSQL databases. The data doesn't always stay consistent across the entire database as it moves around. This affects consistency on multiple servers. The times that transactions are saved differs from the more consistent saving times with relational systems. There are also many different NoSQL products, some with steep learning curves. Most organizations are not big enough to need an unstructured database like NoSQL. The vast majority of databases are still relational databases using SQL.

Popular NoSQL Database management software products include:

- MongoDB ' open-source, uses a weakly typed JSON structure, written in C++
- Cassandra ' developed at Facebook, uses distributed data storage, and is a columner database
- Dynamo ' Amazon proprietary database, allows documents and graphs as data
- CouchDB ' open source, high concurrency with strict consistency, uses JSON to store data and JavaScript as its query language

Data Types

The type of data that can be stored in relational databases is defined ahead of time. This is referred to as data types. Data types are the rules that define what data may be stored and how it is stored. We bring up the topic here because it's important to figure out the correct data type during the design process. Having to make changes to a data type later after the database is developed can result in data loss. Data types also allow for more efficient storage and enable restrictions on data that can be entered into our database; therefore, keeping

our data consistent and accurate. The following are some of the common relational data types we will be using in our course.

String Data Types

CHAR	Fixed-length string from 1 to 255 characters long. The length is fixed at creation.
VARCHAR	Variable-length string from 1 to 65,535 characters long
ENUM	One string from a predefined list of strings
SET	Zero or more values from a predefined list

Numeric Data Types

INT	Integer (whole number)
TINYINT	Integers at a smaller range -128 to 127 used for Boolean 0 and 1
DECIMAL	Numbers with digits after the decimal point

A note about ENUM: ENUM is not part of the SQL-standard and many database systems do not support it. Using a reference table instead of ENUM is advisable. In MySQL, only use ENUM when you're storing distinct, unchanging values sets where you will never need to store additional related info. It should only have more than 2 to 20 items. If there are only 2 choices use tinyint for 1 or 0. Again, even with MySQL, when there may be any changes to the value options in the future use a reference table instead.

Date and Time Data Types

DATE	Dates from 1000-01-01 to 9999-12-31 as YYYY-MM-DD
DATETIME	Combines DATE AND TIME
TIME	Time as HH:MM:SS in 24 hour format
YEAR	Years ranging from 1901 to 2155

Step 4: Relational Databases

Database Management Systems

Databases are more than 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 correct 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 Principles

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.

Submission

Take the **W02 Prepare** quiz in Canvas

You may use this quizlet to help you study the terms.

Useful Links:

- Next Activity: ERD and Datatypes
- Go to: Week Index · Course Home · Canvas