By the end of this session, you should be able to:

- **Explain the differences between hierarchical and relational databases.**
- **Provide examples of Structured Query Language.**
- **Install and test a MariaDB database.**

A **database** is an organized collection of data. The data can be stored in various patterns or files:

- **Hierarchical** or **navigational** databases link the data into lists, or parent/child type paths
- **Relational** databases store the data in pre-defined tables connected by common data elements.

The term **database** was established in the 1960s, around the time commuters started using **disk** and **drum** storage devices. This storage media provided ***random access*** to data, something the available prior media (tape) did not have. As the number and capacity of computers grew in the 1960s, so did the quantity of data being managed.

Interest in standards grew, and the group that was leading the standardization of the **COBOL** computer programming language proposed a database approach that was referred to as **CODASYL**.

This approach relied on data being managed in a **navigational** architecture.

Many variations and task-specific databases exist. To learn more, see https://en/wikipedia.org/wiki/Category:Types_of_databases.

17.4.a. Hierarchical Databases

Hierarchical databases (also known as **navigational**) used linked lists called **`data sets`** to form a large network of data. These linked lists typically had forward and reverse pointers to other members in the data set. The data could be located using one of several methods:

- A primary key, possibly using a hashing algorithm.
- Navigating a relationship (a set of data).
- Sequential scanning of records.

Two leaders in this type of database were:

- **IBM**, with its **Information Management System** or **IMS**
- **Cincom Systems** and its **TOTAL** database remained in production into the 2010 era.

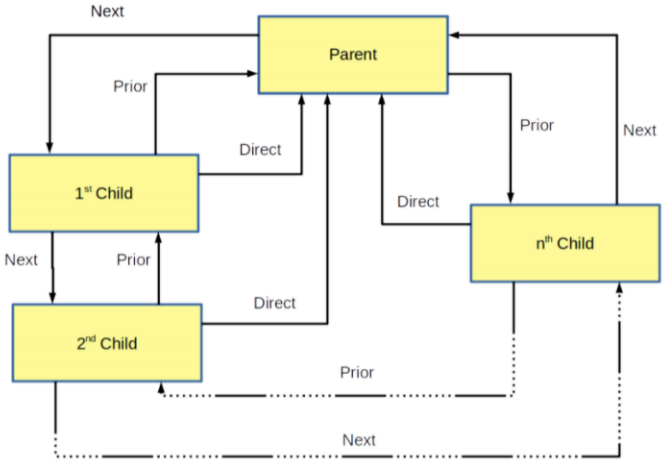

Figure 17.1. Hierarchical Databases

17.5.a. Relational Databases

The **Relational Database** as we know it today grew out of lack of the **Navigational** database's search abilities. This is one of the main contributions to the technology of the relational database.

The original research paper is available at: http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf.

The basic concepts of the **relational database** are:

- Data is organized into tables.
- Tables can be linked to other tables by a common data element.
- Tables can be added, changed and deleted as required.

The **Database Man agent Systems** provide the management of the information in the storage medium and provide an interface to access the information. Probably the most common database interface is **SQL** (**S**tandardized **Q**uery **L**anguage).

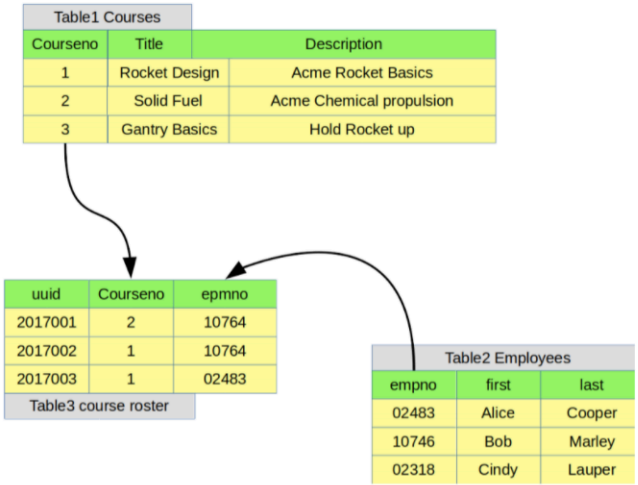

Figure 17.2. Relational Databases

Database Management Systems (**DBMS**) are the programs (managers) that provide access to the data. **DBMS** provide an interface to allow the ***manipulation*** of the information stored. The **Database** is typically referred to by the name of the management system. Some popular **DBMSs** are **MariaDB**, **Oracle**, **MySQL**, **PostgressSQL**, **MongoDB**. The most popular **DBMS** is an active debate; one popularity listing can be found at https://db-engines.com/en/ranking.

Since there are hundreds of **DBMSs** to choose from, the **best** ones usually follow a discussion that ends in "it depends". Here are some discussion points for the few highlighted:

- **MariaDB** (https://mariadb.org)
  RDBMS, open source, freely available, a fork of **MySQL**.
- **Oracle** (https://www.oracle.com/database/enterprise-edition/index.html)
  RDBMS, long history, supported "Enterprise Editions".
- **MySQL** (https://www.mysql.com/)
  RDBMS, started out as an independent open source, now owned by **Oracle**.
- **PostgressSQL** (https://www.postgresql.org/)
  RDBMS, open source, long history.
- **MongoDB** (https://www.mongodb.com/)
  Open source, Document DBMS

Structured Query Language (**SQL**) is one of the first computer languages to be used to access relational databases. It supports data insert, query, update, and delete. It became a standard of the **American National Standards Institute** (**ANSI**) in 1986. Some examples of **SQL** language usage can be found at http://www.itl.nist.gov/div897/ctg/dm/sql_examples.htm.

Below you can see some examples of **SQL** statements.

Add a table:

```
CREATE TABLE Courses (
    -> Courseno INT NOT NULL ,
    -> Title VARCHAR(100) NOT NULL,
    -> Description VARCHAR(200),
    -> PRIMARY KEY ( Courseno )
    -> );
```

Select information from a table:

```
SELECT * FROM Employees ;
```

Insert data into a table:

```
INSERT INTO Employees (Empno,First,Last)
    -> Values (02483,'Alice','Cooper'),
    -> (10746,'Bob','Marley'),
    -> (02318,'Cindy','Lauper');
```

Putting the components together:

- Data is stored in fixed length **tables**. The data definition is created with the table creation and stored with the data. Table clumsy can be added, changed or deleted.
- The tables use data elements to connect to each other in **relationships**. Something like a `part number` could be used in a part description from the supplier and in an invoice table when the part is used by a customer.
- The **RDBMS** (Relational Database management System) ensures the consistency of the data with the data definition. The **RDBMS** assumes the control of the management of the tables and subsequent storage, thereby freeing the user/programmer from the detailed requirement of manipulating files directly.
- **SQL** is used to request add, delete, query, etc. from the **RDBMS**. This standardized interface allows quicker, more robust applications to be created.

Similar to other network servers, **RDBMS** servers listen for questions and commands and respond appropriately.

Although on the surface a basic **RDBMS** architecture seems easy, complex configurations, performance tuning scaling, and availability targets require an in-depth understanding of not only the business application, but also the internal intricacies of the **RDBMS** itself.

As an introduction to databases, we will install, create tables, add data and query a **MariaDB** database.

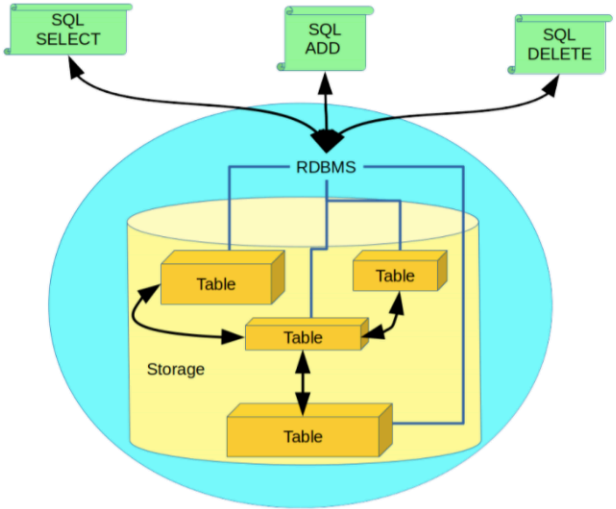

Figure 17.3. RDBMS Components