Database

Data storage is an essential role to all things in the programming world. And as such, it cannot be left in text files sitting in a C:\User\Documents folder. The access time and the parse time would be incredibly high. So everyone that has any kind of software will also have a database set up. Almost all databases use a a flavor of SQL or "Structured Query language". SQL could be considered a programming language of its own so It is impossible to learn it all in a day. But we will go over a few basic concepts as well has how to use them in java.

Also the tutorial on this incredibly broad subject is fairly inadequate for a good understanding. Playing with the SQLDemo project will provide some more insight on how the system works.

Startup

First we are going to use SQLite as our database. This is because it is self contained and does not require downloads. It is a simpler solution and not made for Enterprise Resource solutions, But it is used occasionally in commercial products such as Redcort Virtual Time Clock.

We are going to use a Gradle project to automatically download the Java Database Controller drivers or JDBC for short

add this line to your dependencies

```
implementation group: 'org.xerial', name: 'sqlite-jdbc', version: '3.34.0'
```

To create a database

```
public static Connection createNewDatabase(String fileName) {

    // This is just a file path "jdbc:sqlite" is the driver to use and
C:/sqlite/db is a file path to store the database file at
    String url = "jdbc:sqlite:C:/sqlite/db/" + fliename;

    try (Connection conn = DriverManager.getConnection(url)) {
        if (conn != null) {
            return conn;
        }
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
    return null;
}
```

This is is basically the same as the way you will connect to the database. so you could call it something like createOrConnect.

The following is an SQL tutorial. if you do not need it then skip to the java part all of the SQL will be done in the java program so it is not necessary to go through the short tutorial. There a many tutorials online and books online that go into more detail and give you a better understanding of SQL.

SQL

SQL relies on 4 basic statements

- **INSERT** - Adds records to a table in the database
- **UPDATE** - Updates database records
- **SELECT** - Queries the database and returns records
- **DELETE** - Deletes records from the database.

All data is organized into **tables**. Each row in a table is know as a **record** each table must have a **primary key.** A primary key is simply a unique field. One that will never be shared by any other record. Typically this is an ID column, but it can be whatever you make it as long as all records have a unique Primary key.

Here is the syntax of each command. where items in **[ ]** are optional and items inside **< >** Are determined by the programmer But dont get tied up in these too much. This is just a basic look at the syntax there will be use cases below to give you a better idea

SELECT

```
SELECT <column names> FROM <table name> [JOIN <table2> ON <table1 column> = <table2 column>] [WHERE <condition>]
```

INSERT

```
INSERT INTO <table name> (<column names of columns you want to give values seperated by commas>)  VALUES ( <data>)
```

INSERT can also be pared with a SELECT to look like this

```
INSERT INTO <table name> (<column names>) SELECT <data> FROM <table> [WHERE <condition>]
```

UPDATE

```
UPDATE <table name> SET <column name> = <value>[, <colum name2> = <value2>...] [WHERE <condition>]
```

It is important to note that for **UPDATE** and **DELETE** if you do not limit them with WHERE they will change or delete all records in the table.

DELETE

```
DELETE FROM <table name> [WHERE <condition>]
```

Practical examples using the following table **employee**.

employee_id	fname	lname	hire_date	wage
1	Adam	Sandler	01-01-2018	12.78
2	Carl	Williams	04-06-2018	11.58
3	Cayden	Adams	03-15-2017	14.00
4	John	Green	09-23-2016	16.00
5	Amy	Cooper	12-12-2020	9.75

SELECT

```
SELECT fname,lname,wage FROM employee

| fname  | lname    | wage  |
|--------|----------|-------|
| Adam   | Sandler  | 12.78 |
| Carl   | Williams | 11.58 |
| Cayden | Adams    | 14.00 |
| John   | Green    | 16.00 |
| Amy    | Cooper   | 9.75  |

SELECT * FROM employee WHERE wage < 12.00 -- '*' is short for all columns. if
you have multiple tables you can use table.* for all columns in that table

| employee_id | fname | lname    | hare_date  | wage  |
|-------------|-------|----------|------------|-------|
| 2           | Carl  | Williams | 04-06-2018 | 11.58 |
| 5           | Amy   | Cooper   | 12-12-2020 | 9.75  |
```

UPDATE

```
UPDATE employee SET wage = 10.00 WHERE employee_id = 5; -- WE just gave amy a
pay raise

UPDATE employee SET wage = 17 -- we just gave eveyone a pay raise.
```

INSERT

```
INSERT INTO employee (employee_id,fname,lname,hiredate) VALUES
(6,'Carly','Burton','05-25-2020')
-- This added a new employee where wage is null.
```

DELETE

```
DELETE FROM employee WHERE employee_id = 6 -- this deleted carly record, she
never showed up for her first day.

DELETE FROM employee WHERE fname = 'Carly' -- this deletes every employee with
the first name of carly. which is probably not how you would want to use delete
in this case.
```

Java

Now that we have our database created we are going to create two tables, Sales Rep And Invoice.

salesrep		invoice	
salesrep_id	integer	invc_id	integer
sr_fname	text	invc_number	text
sr_lname	text	invc_sr_id	integer
sr_commission	real	invc_ammount	real

Each invoice has a unique sales rep linked to it by the invc_sr_id column. Sales reps make commission on each sale base on the sr_commission value. this we are going to create the tables and insert some data into the tables. then we will calculate each sales reps commission based on the data in the database.

Create tables

SQLite for both of the table creations

```
CREATE TABLE IF NOT EXISTS salesrep ( salesrep_id integer PRIMARY KEY, sr_fname
text, sr_lname text, sr_commission real);
CREATE TABLE IF NOT EXISTS invoice ( invc_id integer PRIMARY KEY, invc_number
text, invc_sr_id integer, invcammount real);
```

Java code to create the tables will look like this

```
public static void createTables() {
    // Connection url to the database file
    String url = "jdbc:sqlite:c:/sqlite/db/test.db";

    // putting the SQL as a string in java
    String salesrepSQL = "CREATE TABLE IF NOT EXISTS salesrep " +
                         "( salesrep_id integer PRIMARY KEY, sr_fname text,
sr_lname text, sr_commission real);";
    String invoiceSQL = "CREATE TABLE IF NOT EXISTS invoice " +
                         "( invc_id integer PRIMARY KEY, invc_number text,
invc_sr_id integer, invcammount real);";

    try (Connection conn = DriverManager.getConnection(url);
         Statement stmt = conn.createStatement()
         Statement stmt2 = conn.createStatement()) {
            // create a new table
            stmt.execute(salesrepSQL);
            stmt2.execute(invoiceSQL);
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }
}
```

Now that the tables are created lets create a function to add records to the tables

```
public static void addSalesRep(String fname, String lname, double commission) {
    // Connection url
    String url = "jdbc:sqlite:c:/sqlite/db/test.db";

    // SQL where all fields are replaced with '?'
```

```
    String sql = "INSERT INTO salesrep (sr_fname,sr_lname,sr_commission) VALUES
(?,?,?)"

    try (Connection conn = DriverManager.getConnection(url); // make the
connection to the file
          PreparedStatement  pstmt = conn.prepareStatement(sql) // Create a
prepared statement){
        pstmt.setString(1, fname); // Set the first paramater (or '?') to fname
        pstmt.setString(2, lname); // Set the second parameter to lname
        pstmt.setDouble(3, commission); // Set the third parameter to commission
        pstmt.executeUpdate(); // execute the prepaired statement with the set
parameters
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

With this function in place. You no longer need to build an SQL statement when you want to add a sales rep. simply call the function like this

```
addSalesRep("John","Doe",.012);
```

Now that we have covered the basics. I included an SQL file to load a bit of data into the database. I also included a function inside the main project to load and run the SQL file. so inside the main class set the UPLOAD constant to true and run the program. It should terminate shortly after and load up records for the next part which is using the data.

Query and Join

SQL uses **Join** in queries to add tables to the result set. so if we want to get the sales rep records with every invoice we would do a query like this

```
SELECT * FROM invoice JOIN  salesrep ON salesrep_id = invc_sr_id
```

The record you join on is important. It will join them based on matching records and may fail or return false data if you join on records that don't have any meaning. in this case invoice has a record specifically to correlate it with sales rep.

Now lets make a query return the commission from every sales order.

```
SELECT invc_number, (sr_commission * invc_ammount) FROM invoice JOIN salesrep ON
salesrep_id = invc_sr_id
```

That will give us records of invoice number and the amount of commission made on each sale.

If we take this idea a step further and switch around the data sum. we can use the **sum()** function

```
SELECT sr_id, sr_lname, sr_fname, sum(invc_ammount) * sr_commission FROM
salesrep JOIN invoice ON salesrep_id = invc_sr_id
GROUP BY sr_id,sr_lname,sr_fname, sr_commission;
```