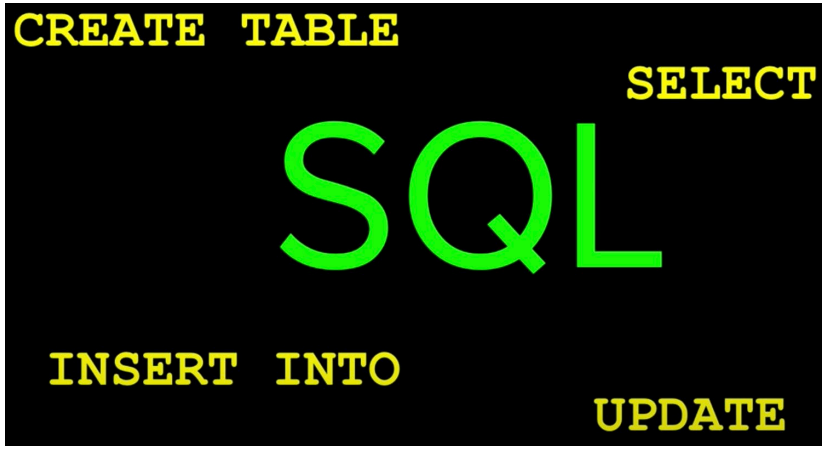
Structured Query Language (SQL)



Database Table

- The **top row** are the column names.
- Each **subsequent row** contains data relating to one thing (in this case, a person).
- Each row should be different, although the same value can appear in different rows of the same column.
- Each row can be uniquely identified by a **Primary key** (a column where no value is repeated).

Num	Name	Inaugural_Age	Age_at_Death
1	George Washington	57.2	67.8
2	John Adams	61.3	90.7
3	Thomas Jefferson	57.9	83.2
4	James Madison	58.0	85.3
5	James Monroe	58.8	73.2
6	John Quincy Adams	57.6	80.6
7	Andrew Jackson	62.0	78.2
⋮	⋮	⋮	⋮

Fig 1. A relational database table, Presidents

DDL

- The DDL to create this table in a database looks like this:

```
CREATE TABLE Presidents (
Num Int PRIMARY KEY,
Name VARCHAR(10),
Inaugural_Age DOUBLE,
Age_at_Death DOUBLE
);
```

More DDL

- Delete the table (completely):
  - ```
    DROP TABLE Presidents
    ```

- Change the columns or other details of the table:
  - ```
    ALTER TABLE Presidents ADD Email VARCHAR(255);

    ALTER TABLE Presidents DROP COLUMN Email;
    ```

DML

- These commands add data to tables, delete it, modify it or get data from the table :
  - `SELECT Name FROM Presidents`
- Gets the contents of the name column.
  - `SELECT * FROM Presidents;`
- Gets all of the columns.

More DML

```
INSERT INTO Presidents(Num, Name)
VALUE (8, Jimmy Carter);
```

- Adds a new row to the table, with an *8* in the **Num** column and *Jimmy Carter* in the **Name** column.
- Note that we can leave some cells (intersection of row and column) empty.

DML --- WHERE

- The **WHERE** command limits the action of the rest of the command to specific rows.
- Does a logical test on each row and if the logic returns TRUE, performs the action.
  - ```
    SELECT * FROM Presidents
    WHERE Name = 'Jimmy';
    ```
- Delete all the rows where Age_at_Death is larger than 90.
  - ```
    SELECT * FROM Presidents
    WHERE Age_at_Death > 90;
    ```

DML --- WHERE Logic

- We can use boolean logic to combine **WHERE** conditions.
- To remove all of the rows containing the age at death around 80 to 90.
  - ```
    SELECT * FROM Presidents
    WHERE Age_at_Death > 80 AND Age_at_Death < 90;
    ```
- To remove only the row containing the age at death larger than 90 with the inaugural age smaller than 60.
  - ```
    SELECT * FROM Presidents
    WHERE Age_at_Death > 90 AND Inaugural_Age < 60;
    ```

SQL Injection and Prevention

- SQL injection covers a range of database attacks from the injection of exploit code through a buffer overflow in a DBMS to execution or arbitrary SQL script through a web page form.

  - Example: http://www.unixwiz.net/techtips/sql-injection.html
  - Documentation: http://msdn.microsoft.com/en-us/library/ms161953.aspx

SQL Injection and Prevention

- Problem occurs because lazy programmers pass un-sanitized user-input directly into SQL command strings.
- Can be prevented by:
  - Sanitizing:
    - using **Trim()** and **Replace()** (asp) to remove escaping and long strings
    - sanitising with regex (php: **preg-replace()**, asp: **rewrite**)
    - remove or escape these characters: **' ; - " ( ) = / #**
  - Passing parameters to DBMS
  - Using stored procedures on the server
  - **mysql_real_escape_string()**, **addslashes()**, **htmlencode()**

SQL Parameters

- **Wrong:**

```
Dim SQL As String = "SELECT Count(*) FROM Users
                    WHERE UserName = '" &  username.text & "’
                    AND Password = '" & password.text & "'"
Dim thisCommand As SQLCommand = New SQLCommand(SQL, Connection)
Dim thisCount As Integer = thisCommand.ExecuteScalar()
```

- **Right:**

```
Dim thisCommand As SQLCommand = New SQLCommand("SELECT Count(*) " &
    "FROM Users WHERE UserName = @username AND Password = @password",
    Connection)
    thisCommand.Parameters.Add ("@username", SqlDbType.VarChar).Value =
    username
    thisCommand.Parameters.Add ("@password", SqlDbType.VarChar).Value =
    password
Dim thisCount As Integer = thisCommand.ExecuteScalar()
```

SQL Stored Procedures

- **Code running on the DBMS**

```
Dim thisCommand As SQLCommand = New SqlCommand
("proc_CheckLogon", Connection)


thisCommand.CommandType = CommandType.StoredProcedure
thisCommand.Parameters.Add ("@username", SqlDbType.VarChar).Value =
username
thisCommand.Parameters.Add ("@password", SqlDbType.VarChar).Value =
password
thisCommand.Parameters.Add ("@return", SqlDbType.Int).Direction =
ParameterDirection.ReturnValue
Dim thisCount As Integer = thisCommand.ExecuteScalar()
```

Ref: SQL Injection Prevention Cheat Sheet
https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

SQL Logic Attack

- Inject <something> OR <TRUE>
- x' or 'x' = 'x
- 1 OR 1=1 * //MySQL (comments: out the rest of the SQL/php)
- A' OR 2=2 ; -- //Other DBMSs

- Solution: Sanitize, filter, restrict privileges

SQL Logic Attack

- UNION
- Concatenates two DML queries
- As long as the number of columns returned is the same.
- MySQL only

- Solution: Sanitize, filter, restrict privileges