SQL Basics

- SELECT - extracts data from a database
  -> select ...from ... where
  -> select distinct...from...(without duplicates)
  -> select count(distinct ...) from ...
  -> select count(*) as distinct.. from (select distinct ... from ...)
  -> select..from..order by...desc(aufsteigend)/asc(absteigend)
  -> select...from..where..is null/is not null
  -> select min(column_name) as minimum from table_name
where

- UPDATE - updates data in a database
  - Update table...set column1=value1,...where condition
- DELETE - deletes data from a database
  - Delete from...where
  - Delete from table_name (delete all rows in table)
- INSERT INTO - inserts new data into a database
  - Insert into (columns) values (value, value,...)
- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database
- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- CREATE INDEX - creates an index (search key)
- DROP INDEX - deletes an index
-

SQL Injections

- Based on „=“ ist always true
  - like SELECT * FROM Users WHERE userId = 105 OR 1=1 -> 1=1 always true so all information regarding the user will be displayed
  - Enter in text field for user name or password field: **" or ""="** -> always true
- Based on batchedSQL Statements
  - Like SELECT * FROM Users WHERE UserId = ‘‘ + txtUserId;
    - Enter in text field for user name : 105; DROP TABLE blablabla -> will delete complete table suppliers
- Protection: SQL Parameters
  - Parameters represented bei a @ marker
  - UserID@0 ‘ ‘ ; checks each parameter and treats it literally - not as

sql statement to be executed
    - E.g : txtSQL = "INSERT INTO Customers (CustomerName,Address,City) Values(@0,@1,@2)";
    - In ASP.NET web language:
    - txtUserId = getRequestString("UserId");
      sql = "SELECT * FROM Customers WHERE CustomerId = @0";
      command = new SqlCommand(sql);
      command.Parameters.AddWithValue("@0",txtUserId);
      command.ExecuteReader();

Vorbereitung Begriffe

- Operator Union
    - combines the result set of two or more SELECT statements (only distinct values)
        - e.g.:

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;
```

    - UNION ALL combines the result set of select statements (allows duplicates)

- CONCAT_WS Function
    - Add several expressions together and add a ‘‘-‘‘ separator between them

```
SELECT CONCAT_WS(“-“, “SQL“, “Tutorial“, “is“, “fun!“) AS ConcatenatedString;
```

    - CONCAT_WS(separater, Expression1, expression2,...) -> if seperator is NULL, function returns NULL
    - SELECT CostumerName, CONCAT_WS(“ “, Adress,PostalCode, City) AS Adress FROM Costumers; —> returns table with columns CostumerName and Adress(e.g Obere Straße 57 12209 Berlin)

- GROUP_CONCAT Function
    - Returns a string with concatenated non-NULL value from a group
    - Return NULL when there are no non-NULL values
    - Syntax: GROUP_CONCAT(expr);
    - Example: SELECT pub_id, GROUP_CONCAT(cate_id) FROM book_mast GROUP BY pub_id; —> will return a list of comma

separated ‘cate_ids‘ for each group of ‘pub_id‘ from book_mast table

- INFORMATION_SCHEMA
  - Database within each MySQL instance -> stores information about all the other databases that the MySQL server maintains -> read-only tables ->views not base tables-> no files associated with them -> no database directory with that name
  - INFORMATION_SCHEMA is usable as default database but you can only read contents, not perform INSERT, UPDATE, DELETE

```
SELECT table_name, table_type, engine
  FROM information_schema.tables
  WHERE table_schema = 'db5'
 ORDER BY table_name;
```

  - Queries that search for information in more than one database might take a long time and impact performance -> to check this use EXPLAIN

- Comments
  - Used to explain sections of SQL statements or to prevent execution of SQL statements
    - Single line comments: - - ignores anything after the symbol and the end of the line
    - Multi-line comments: /* */