SAVONIA

EFA8040
Data Management and SQL

SQL, part 1

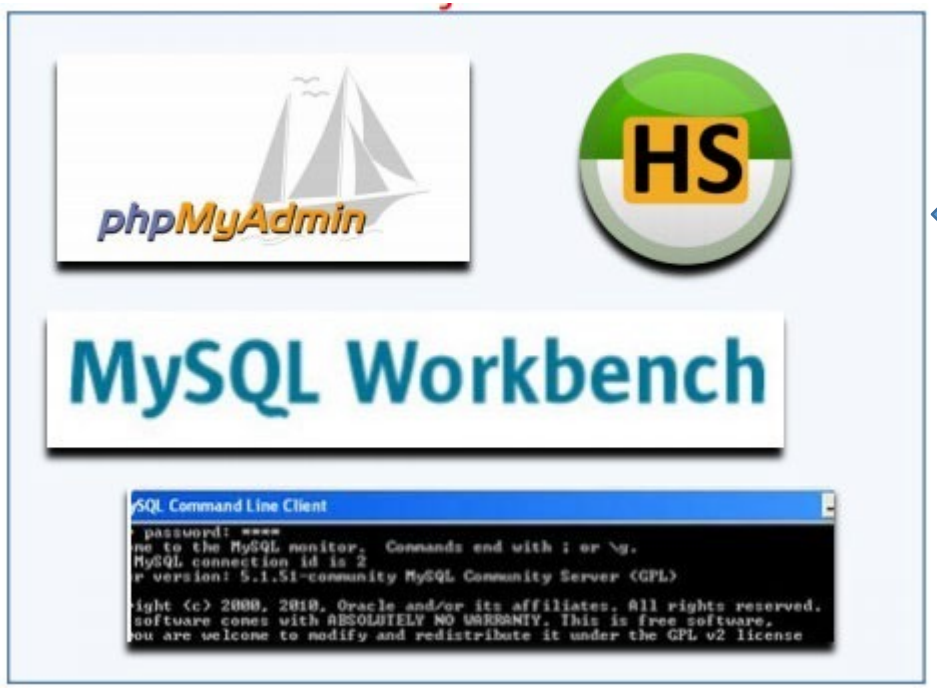
Markku Kellomäki,
markku.kellomaki@savonia.fi

SQL language

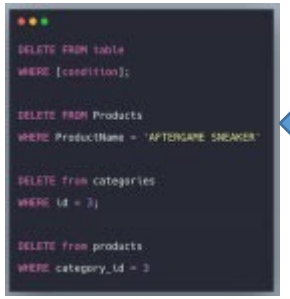
- SQL (structured query language) is a programming language that enables programmers to work with that data
- SQL language is declarative, non imperative programming language
  - Imperative languages are for example Java, JavaScript, C#, C++
  - With SQL you can execute calculations, but not define how task will be executed
- There are programming languages that are imperative and build on top of SQL like PL/SQL and T-SQL
- You can execute SQL language commands
  - Manually with query interface
  - As a part of some other software(”embedded SQL”)

Examples of SQL-tools

SQL-clients



Applications



SQL

SQL

SQL-database software

SQL language commands

SQL languages commands can be divided to different groups

QUERY LANGUAGE

- Request query of tables or views of database
- Query returns a set of rows
- Query doesn’t alter databases data content
- Command: `Select`

DML commands

DATA MANIPULATION LANGUAGE (DML)

- Are used for managing data within tables
- Commands: `INSERT, UPDATE, DELETE`
- DML commands form transactions which can be accepted or denied

DDL commands

DATA DEFINITION LANGUAGE (DDL)

- DDL statements or commands are used to define and modify the database structure of your tables or schema
- For example creating, deleting or altering tables
- Commands: `CREATE, ALTER, DROP`
- DDL commands cause transactions to end

SAVONIA

Basic queries

Select

SELECT query syntax

`SELECT`	what columns are wanted
`FROM`	from what tables/views the data is fetched
`WHERE`	what rows are fetched
`GROUP BY`	how data is grouped
`HAVING`	which rows will be fetched
`ORDER BY`	how data is sorted

- Order of parts of commands is important
- `SELECT` and `FROM` are mandatory

SQL syntax

- SQL queries are not case sensitive
- With some cases in the names of objects case matters
- Queries end to a semicolon ;

```
SELECT * FROM table;
select * from table;
```

--same query

```
Select * from table;
Select * from Table;
```

--can mean different tables

SQL syntax

- The data usually is case sensitive

```
Select fname, lname, city from
     person  where city='KUOPIO';
Select fname, lname, city from
     person  where city='kuopio';
```

- Depending on settings queries can return different rows. Case sensitivity can be adjusted by changing collate settings.

Executing SQL queries

- When executing MySQL/MariaDB queries with SQL-client (HeidiSQL, command line editor, MySQL Workbench), set your database as default or execute command ”Use mydatabase;”
- After this queries will be directed to that database
- Queries to specific table is done with command ”`select * from mytable;`”
- If you want to use data from another schema, use that schemas name by adding it to front of table like this ”`select * from mydatabase.mytable;`”

Limit rows

- You can limit result of rows by setting limit

```
select fname, lname from person
limit 3
```

- Works with MySQL and MariaDB

Select only specific columns

- `select * from person;` selects every row with every column

- `select fname, lname from person;` selects only fname and lname columns.

Using aliases

- You can use aliases to change result of names of columns using `AS`
- It doesn't change the name of column, *alias shows only in result*

```
Select fname AS firstname, lname as lastname from
person;
```

- If there's space in alias, put the alias in ' ' (straight apostrophes)

```
Select fname AS 'first name', lname as 'last name' from
person;
```

Distinct

- Distinct removes any duplicates from results

```
Select city from person;
```

Vs.

```
Select distinct city from person;
```

Where

- `WHERE` limits what rows will be fetched
- Strings must be in quotes, others not

```
Select fname, lname, city  From person
Where city = 'TAMPERE';

Select fname, lname, city  From person
Where salary > 3000;
```

Operators in the Where Clause

`=`	Equal to
`>`	Greater than
`<`	Less than
`>=`	Greater than or equal to
`<=`	Less than or equal to
`<>`	Not equal to
`!=`	Not equal to
`BETWEEN`	between a certain range
`LIKE`	Search for a pattern
`IN`	to specify multiple possible values for a column

Operators in the Where Clause

`NULL` means only that the value is missing. You can only compare nulls with these:

`IS NULL`	if value is missing
`IS NOT NULL`	if value exists

Combining operators happens with `AND`– and `OR`-operators

`AND`	both must be true
`OR`	one or other must be true

Comparison

- All comparisons and sorting are based on database's capability to order comparable values
- Ordering works with basic data types
  - Numeral data types => order of magnitude
  - Strings => alphabetical order
- For example it is impossible to compare objects without any information about the way to compare. Some SQL-databases offer support to code comparison function

Operators <, <=, =, >, >=

- Fetch persons with salary greater than or equal to 2800

```
Select fname, salary
From person
Where salary >= 2800;
```

- Fetch names of people with the last name starts with the letter 'K' or follows it in alphabets

```
Select lname,
salary  From
person
Where lname >= 'K';
```

Operators <>, !=, Not

- Fetch persons whose salary is not 2800

```
Select lname, salary
From person
Where salary <> 2800;

Select lname, salary
From person
Where salary != 2800;

Select lname, salary
From person
Where not (salary = 2800);
```

NULL comparison

- NULL is uncomparable with =, >, < operators, because NULL is not a value
- NULL means that the value is missing

```
SELECT lname,degree
From person
Where degree IS NULL;

SELECT lname, degree
From person
Where degree IS NOT NULL;
```

Multiple conditions

- If there are multiple conditions, `AND` or `OR` operators are used to create logical connections between conditions.
- `AND` both contitions must be true;
- `OR` at least one condition must be true

```
Select fname, lname, salary, city
From person
Where city = 'TAMPERE' AND salary < 2700;
```

Multiple conditions

```
Select fname, lname, salary, city  From person
Where city ='TAMPERE'  OR salary < 2700;

Select fname, lname, city, degree  From person
Where city ='HELSINKI' OR degree is NULL;
```

Multiple conditions, order of comparison

- Comparisons are done in order in pairs:
  - EXCEPT if there are parenthesis, where comparison in parenthesis is done first)
  - EXCEPT if there are multiple comparisons and `AND` and `OR` operators are combined
    - `AND` operators are compared first, then `OR`
    - Compare to mathematics: Multiplication is done before addition

Multiple conditions, order of comparison

```
SELECT lname, salary, city,taxperc
From person
Where (taxperc = 22 OR taxperc =37)  AND salary = 2800;
```

Compare with this:

```
SELECT lname, salary, city,taxperc
From person
Where taxperc = 22 OR taxperc =37  AND salary = 2800;
```

So if tax percentage is 22, salary doesn't matter because OR requires only one to be true

IN operator

- IN-operator lists all values that meets its demand

```
Select fname, lname, city
From person
Where city in ('TAMPERE', 'TURKU');
```

NOT IN operator

- Fetch all who do not work in departments 1 and 3

```
SELECT fname, lname, depcode
From person
Where depcode not in (1,3);
```

- Fetch all whose degree is not 'MSc'

```
SELECT fname, lname, degree
From person
Where degree not in ('MSc');
```

BETWEEN operator

- BETWEEN operator fetches value by limiting with upper and lower limits.
- Fetch all whose tax percentage is between 22 and 30.

```
Select fname, lname, taxpers
From person
Where taxperc between 22 and 30;
```

- Fetch all whose last name is between Aalto and Junkkari

```
Select fname, lname
From person
Where lname between 'Aalto' and 'Junkkari';
```

Operator NOT BETWEEN

- NOT BETWEEN-operator fetches values below lower limit and over upper limit
- Fetch all whose tax percentage is not between 24 and 31

```
Select fname, lname, taxperc
From person
Where taxperc NOT BETWEEN 24 and 31;
```

LIKE operator

- LIKE operator fetches data based on pattern
- Wildcards
  - _ replaces one character
  - % replaces 0-n characters

- Fetch all projects which location starts with letter H

```
Select pname, location
From project
Where location like 'H%';
```

LIKE operator

- Fetch all persons whose last name has 'k' in it

```
Select lname
From person
Where lname like '%K%';
```

- Fetch all persons whose last name has a letter 'o' as a second character in last name

```
Select lname From person
Where lname like '_o%';
```

ORDER BY

- The `ORDER BY` keyword is used to sort the result-set in ascending or descending order.

```
Select lname, salary  From person
Order by salary;
```

- The `ORDER BY` keyword sorts the records in ascending order by default. To sort the records in descending order, use the `DESC` keyword.

```
Select lname, salary  From person
Order by salary desc;
```

- `NULL` is not orderable

Order by several columns

```
Select lname, fname, salary, city
From person
Order by salary desc, city, lname, fname;

Select lname, fname, salary, city
From person
Order by 3 desc, 4, 1, 2;
```