1 SQL: Language characteristics

SQL is short for **Structured Query Language**

- Language to operate databases and provides an interface with RDBMS (**Relation Database Management System**)
  - Database creation
  - Data loading
  - Data modification
- SQL has 3 different types of instructions
  1. Data Control Language
     - Grant, Revoke
     - Is concerned with acces privilige and determines who can control which part of the database
  2. Data Manipulation Language
     - Select, insert, update, delete
     - Data retrieval, removal and transformation
  3. Data Definition Language
     - Create, drop, alter
     - Creation and maintenance of databases, tables, views, indices
- SQL is based on the relational model of data
  - Tables, 2D datastructes consisting of rows and columns
  - Rows (instance/record) is a collection of different characteristics that all belong to the same datapoint
  - Column (attribute) is a collection of data describing one specific characteristic over different instances
  - Field is the intersection between a column and a row
  - Primary key, a unique identifier that defines an instance
  - Foreign key, a connecting key between two different tables

2 Selecting data

2.1 SELECT command

The **SELECT** statement allows us to specifiy which data we wish to call from the database in order to view, manipulate, etc. The general synthax for the SELECT statement can be written as

```
1    SELECT [Predicate] {* | table.* | [table.]field1 [, [table.]
     field2 [,...]]} FROM tableexpression
```

The SELECT statement consists of different components

- **Predicate**, a signifier that lets us control which records are displayed
  - **ALL** takes all the data that is present in the ranges we specify
  - **DISTINCT** returns only an overview of the possible values in the
- Field or table specifiers where we can control which tables or which section of a table we wish to select
  - *****, denotes to select everything
  - **tableName**
  - **tableName.fieldName**
  - multiple tables and/or fields can be selected at once using a **,** as a seperation
- **FROM** denotes where our selection comes from
  - table
  - tableexpression, linking multiple tables to each other

2.2 WHERE command

The **WHERE** works as a filter that allows us to retrieve data that needs to fulfill a specific condition or multiple conditions

```
1    WHERE attribute operator constant | attribuut
```

The WHERE statement consists of different components

1. **Attribute** determines which column we will be filtering
2. **Operator** specifies the condition that needs to be fulfilled
   - =, equality operator
   - $<, >, <=, >=$, greater/lesser than
   - $<>, != $, not equal operator
   - BETWEEN, the range operator
   - LIKE, pattern operator using the wildcard operators '%' and '_'
     - '%', represents multiple characters (or nothing) that can be anything
       - 'c%' with c as any character searches for anything starting with the character c
       - '%c%' with c as any character searches for anything that contains the character in the entry
       - '%c' with c as any character searches for anything that ends with the character c

  - '_' represents a single character (or nothing) that can be anything
    * 'c_' with c as any character searches for anything starting with the character c followed by one other character due to the single underscore
    * '_c_' with c as any character searches for anything that contains the character c between two characters due to the single underscore before and after the letter
    * '_c' with c as any character searches for anything that ends with the character c following a single character
    * A repetition of multiple underscores looks for anything that has the same length as the number of underscores

The WHERE statement can be combined with logical operators

- AND, requires both conditions to be fulfilled
- OR, requires one of the conditions to be fulfilled
- NOT, requires the condition not to be fulfilled

2.3 ORDER BY command

ORDER BY command imposes an ordering upon the data retrieved by the query. When using ORDER BY the collation, i.e. differentiation between smaller and capital letters, is dependant on the setting. As a standard mysql does not differentiate between capital letters and lowercase letters.

```
1     ORDER BY attribuut asc|desc
```

- Ordering always happens based on the values of an attribute
- asc orders the values from smallest to largest (ascending). When the values are strings the ordering happens from A to Z
- desc orders the values from largest to smallest (descending). When the values are strings the ordering happens from Z to A

Sorting on mulitple attributes is possible, it should be noted that sorting happens in a sequential fashion. Changing the order changes the output of the query.

2.4 AGGREGATE functions

AGGREGATE functions performs specific computations on a given variable where all recors that are non NULL are taken into account

- COUNT, gives a frequency distribution of each value
- SUM, sums up all the entries

- AVG, averages out over all the entries
- MAX/MIN, finds the upper and lower value in the parameter

```
AGGREGATE(paramName) [as ALIAS]
```

- Similar syntax as functions in programming language where the AGGREGATE is one of the functions mentioned above

as ALIAS , optional syntax that allows us to rename the AGGREGATE to an alias of choice

2.5 GROUP BY command

GROUP BY is used to cluster rows that fulfill a specific condition together which allows us to perform AGGREGATE functions on subsets of the data instead of the data in its entirety. A commonly used keyword in the GROUP BY command is the HAVING keyword. The HAVING keyword functions as a filter but is not equivalent to the WHERE keyword

- WHERE
  - used to filter the records from the table based on the specified condition.
  - can be used without GROUP BY Clause
  - implements in row operations
  - cannot contain aggregate function
  - can be used with SELECT, UPDATE, DELETE statement
  - is used before GROUP BY Clause
  - is used with single row function like UPPER, LOWER etc.
- HAVING
  - is used to filter record from the groups based on the specified condition
  - cannot be used without GROUP BY Clause
  - implements in column operation
  - can contain aggregate function
  - can only be used with SELECT statement

```
SELECT [AGGREGATE]colName(s) FROM tableName WHERE condition(s)
GROUP BY colName(s) HAVING condition(s) ORDER BY ASC | DESC
```

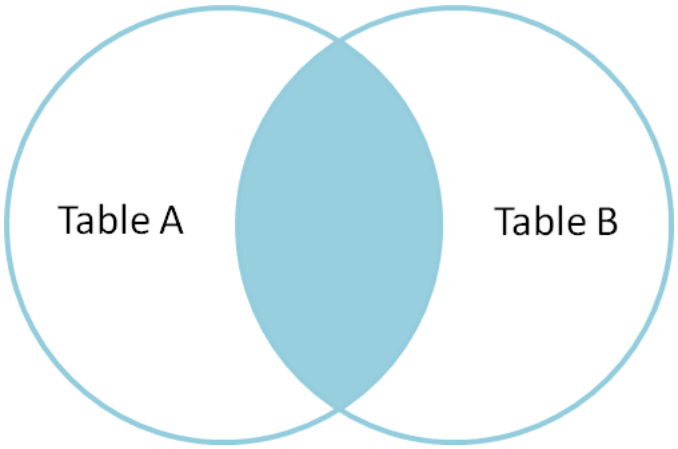
- GROUP BY should always be placed in front of ORDER BY

- If ORDER BY is not used the result will be sorted based on the columns of GROUP BY
- When using the GROUP BY statement then we need to include all columns that are not aggregated to be mentioned in the GROUP BY clause

3 Working with mulitple tables

3.1 Inner join

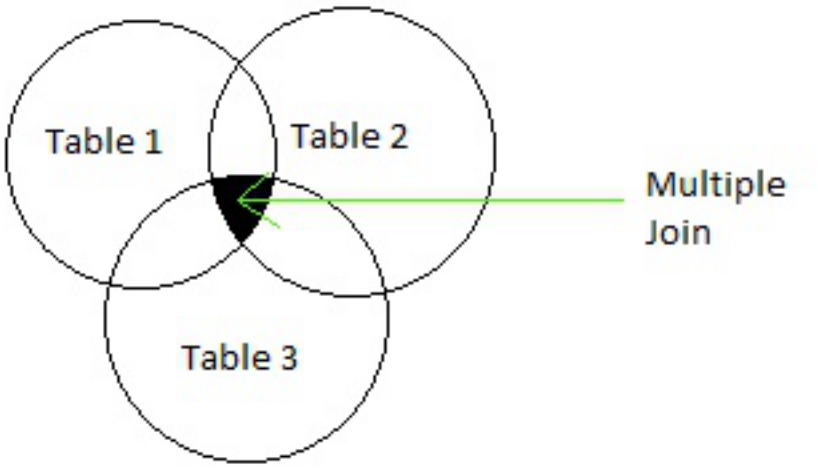
An inner join is a way of combining multiple tables together where the tables display overlap between their data. The inner join can be visualized with the following diagram



Only those rows/records that are matching in both the tables will be included the formation of a merged table. The inner join is used by the command

```
select cols from table_1 inner join table_2 on col_condition(
t1_col operator t2_col) [inner join table_3 on col_condition]
```

- The condition on which we decide which data is to be included is determined by a comparison between the data of both table. Valid operators are '=', '<', '>', '<=', '>=', '! = or <>'
- Multiple conditions can be formulated for the inner join using the AND or OR keywords
- Mulitple tables can be joined simultaneously by following an inner join with another. This results in a new table in which we look for data that is overlapping between multiple tables



3.2 Outer join

Outer join allows us to combine 2 tables together based on a specific condition similarly to how the inner join was used. The major difference between the inner and outer join operations is that outer join allows us to include non overlapping data into the end result of our query. The outer join can be differentiated using the keywords left and right

- left outer join takes the non overlapping data from the first table
- right outer join takes the non overlapping data from the second table

The outer join operation can be represented through the following diagram

```
1    select cols from table_1 [left | right] (outer) join table_2 on
      col_condition
```

- Outer is an optional keyword in this case and does not need to be included into the query

Outer joins and inner joins can be combined together by nesting the joining operations. However **an outer join can not be followed by an inner join**. Despite the fact that this is possible to do in SQL and it will not result in a query error, the result that is generated will be faulty. This is due to the nature of the joining operations that are used.

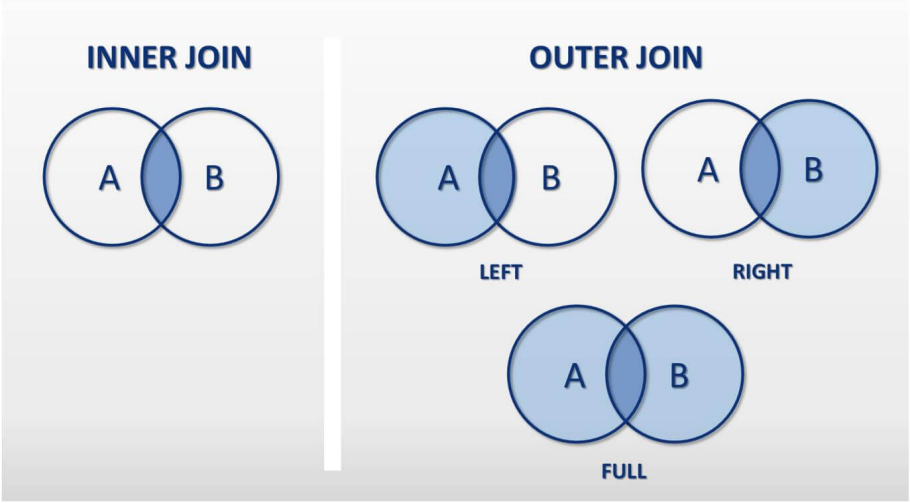


Figure 1: Different outer join operations contrasted against the inner join. Outer join adds upon the inner join by taking the data that is not present in the overlap between tables

- Outer join specifically includes **non overlapping data** in the query result where as the inner join will **only return that information which is overlapping between tables**
- Using an inner join **after** an outer join will **negate the whole point of the outer join**

3.3 Self join

The self join allows us to combine a table with it self

```
1    select cols from table as t1 join table as t2 on t1.col
     operator t2.col
```

- The table has to have a 2 references in order for this to work

4 Mulitple queries

4.1 Union

Multiple queries can be combined into a single result by using the union statement

```
1    query_1 union query_2 ... union query_n
2    query_i: select ...
```

- Each query in the union needs to select the **same amount fields or columns**. The fields do **not need to have the same amount of records or rows**
- Only the first query can be given an alias
- order by can be used but needs to be included at the end of the union.
- If the same fields are in mulitple queries then they will only be returned once. You can force the display of duplicate results using the union all command.

4.2 Subqueries

It is possible to incorperate queries into another query, i.e. multiple selects into a single query

- where select ...
- having select ...
- select select ...
- from select ...
- Insert, update, delete

A special kind of subquery is the **correlated subquery**. This subquery is a query where we refer to a table that appears **in the outer query**. An example of a correlated subquery is

```
1    select * from t1 where c1 any (select c1 from t2 where t2.c2=t1
     .c2)
```

It is important to remember that SQL **evaluates from inside to outside**

5 Modifying tables and data

5.1 INSERT command

The insert command allows us to append a new row to an existing table. The command is build up with the following syntax

```
1    insert into tablename [(col1name, col2name, \dots)] values (
     col1val, col2val, \dots)
```

- The target columns is not required when using the insert statement but it is advised. When not including the the names of the columns the insert function will take the order of the columns as it is stored. If we give an ordering of the columns in the statement then the values will be appended in the same order as our column order

- Column names are placed between round brackets after the table name
- Depending on the data type we use a different formating
  - Quotation marks are used for non numerical data
  - Different values are seperated with a comma
- Mulitple values can be appended at once. These are added by seperating each row of values with round brackets

```
1        insert into tablename [(col1name, col2name, \dots)]
    values (col1val, col2val, \dots), (col1val, col2val, \dots
    ), \dots
2
```

Values from other tables can be appended to an existing table with the insert statement. When extracting data from a different table we use a select query for the colvalues

```
1    insert into tablename1 [(col1name, col2name, \dots)] select
    tablename2.colname from tableexpression where \dots
```