Tomas Sandnes
tomas.sandnes@kristiania.no

Week 2

# DB1102 / PGR 111 – DATABASES

# Today's topics

*(Today's chapters: 2 in Norwegian Book, 3 in English book)*

- Terminology for relational databases
- MySQL Workbench settings you should change
- More SELECT:
  - Data types & NULL
  - DISTINCT
  - GROUP BY & HAVING

# Running an sql file

- An sql file contains a collection of SQL statements.
  - Often used to create the tables in a schema / database.
  - Or to populate a database with data (some hundreds or thousands of rows).

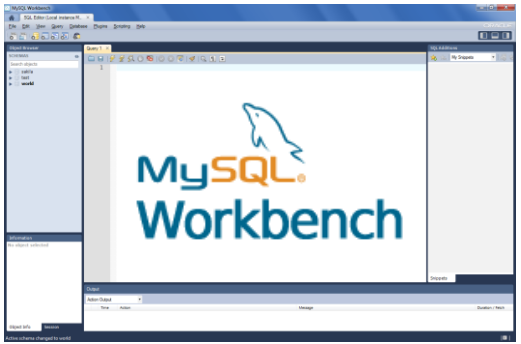
- The easiest way to run the content of these files might be to cut and paste it into the Workbench query window.
  - But there's also an option in Workbench to run an sql file, in: *File -> Run SQL Script…*

- *Note:* Sometimes there will be sql files as part of the lessons or exercises.

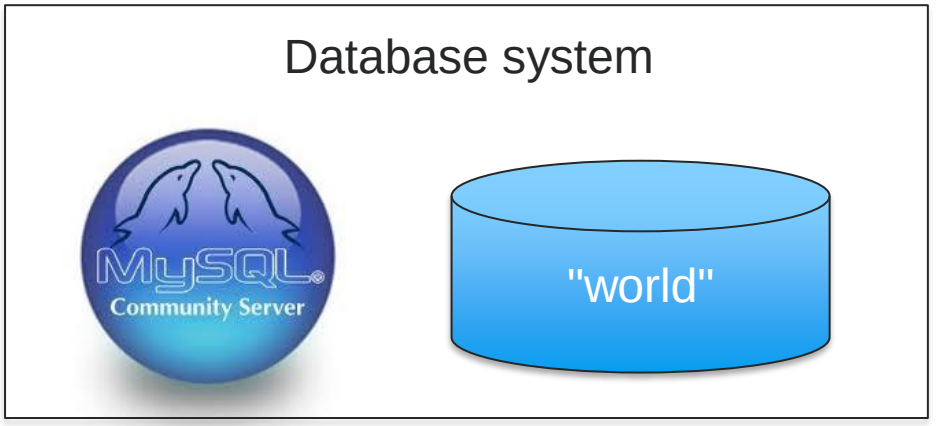
# Terminology

# Database system, repetition

- Database system = DBMS + database ("schema")
  - DBMS (DataBase Management System) => MySQL Server
  - database => "world"




SQL



# The relational database: terminology

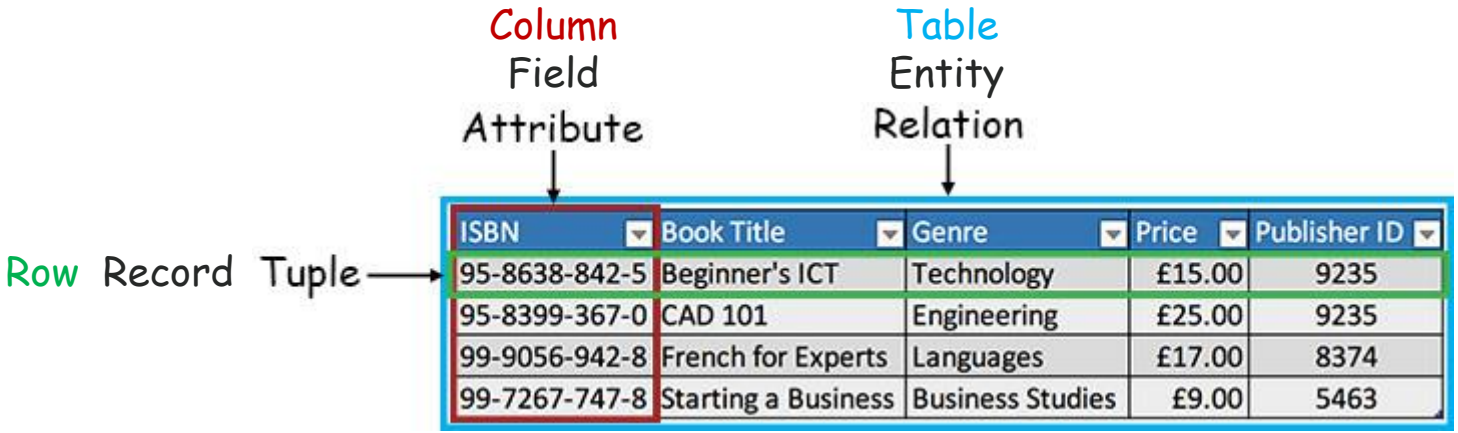
- Relation, *also called entity*, is another word for a table (with columns and rows).
  - Tuple, *also called record*, is another word for a row in the table.
  - Attribute, *also called field*, is another word for a column in the table.

- Domain is all the possible allowable values for an attribute.

- Cardinality is how unique an attribute is in terms of its data values. *Examples:*
  - The primary key field will have a completely unique value for every record. Where there is a large percentage of unique values, this is known as “High Cardinality”.
  - Where there are a lot of repeated values across the entities' tuples, this is known as having “Low Cardinality”.

# The relational database: terminology – cont.

- Some of the terminology, shown with a figure:




| ISBN | Book Title | Genre | Price | Publisher ID |
|---|---|---|---|---|
| 95-8638-842-5 | Beginner's ICT | Technology | £15.00 | 9235 |
| 95-8399-367-0 | CAD 101 | Engineering | £25.00 | 9235 |
| 99-9056-942-8 | French for Experts | Languages | £17.00 | 8374 |
| 99-7267-747-8 | Starting a Business | Business Studies | £9.00 | 5463 |

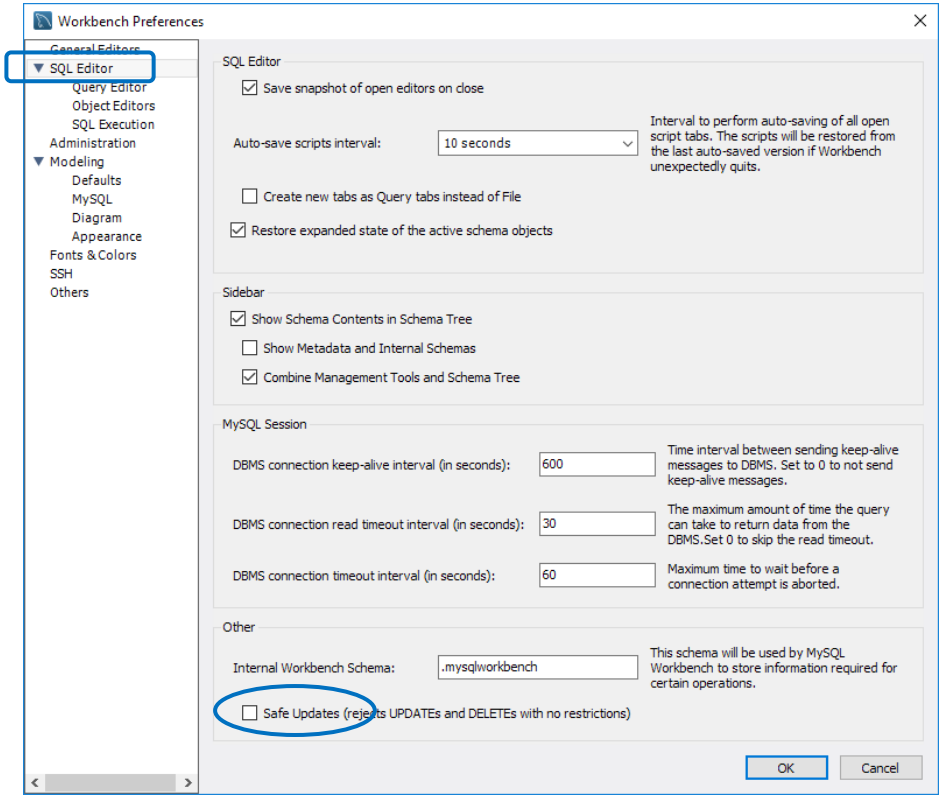
Source: knowitallninja.com/lessons/relational-database-terminology/

# Workbench settings

# MySQL Workbench, settings

IMPORTANT workbench settings:

- Menu: Edit -> Preferences...
  - then SQL Editor
- *Remove mark* on Safe Updates.
  - You may need to scroll down to see this setting in your preferences window.
- Keeping mark here will hinder us from performing some of the exercises coming later in this course.

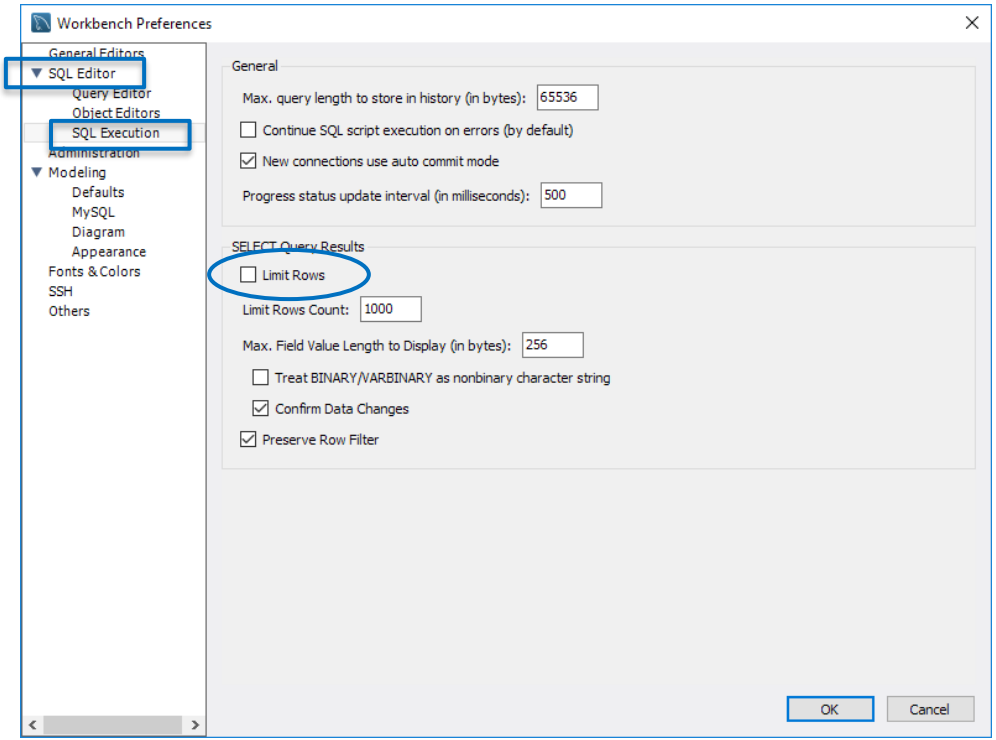


# MySQL Workbench, settings – cont.

- Menu: Edit->Preferences…
  - then SQL Editor
  - then SQL Execution

- *Remove mark* on Limit Rows.

- Keeping mark here will hinder us from getting more than 1000 rows returned per query.



# More SELECT

# A basic SELECT statement, repetition

- A SELECT operation ("query") targets one or more tables.
  - The result is also presented in table format.

```
SELECT Name, Population
FROM city
WHERE CountryCode = 'NOR'
ORDER BY Population DESC;
```

| Name | Population |
|---|---|
| Oslo | 508726 |
| Bergen | 230948 |
| Trondheim | 150166 |
| Stavanger | 108848 |
| Bærum | 101340 |

# Some SQL functions, repetition

- SQL has some built in functions: ("premade logic")
  - COUNT(*) → gives number of rows
  - AVG(column_name) → the average column value of all selected rows
  - SUM(column_name) → the column sum of all selected rows
  - MIN(column_name) → the column minimum value of all selected rows
  - MAX(column_name) → the column maximum value of all selected rows.

- To get easier-to-read results, we can give out-columns aliases by using the keyword AS (or ALIAS).

```
SELECT COUNT(*) AS 'City count'
FROM City;
```

| City count |
| --- |
| 4079 |

# Datatypes

- Name and format for datatypes varies a bit from database to database.

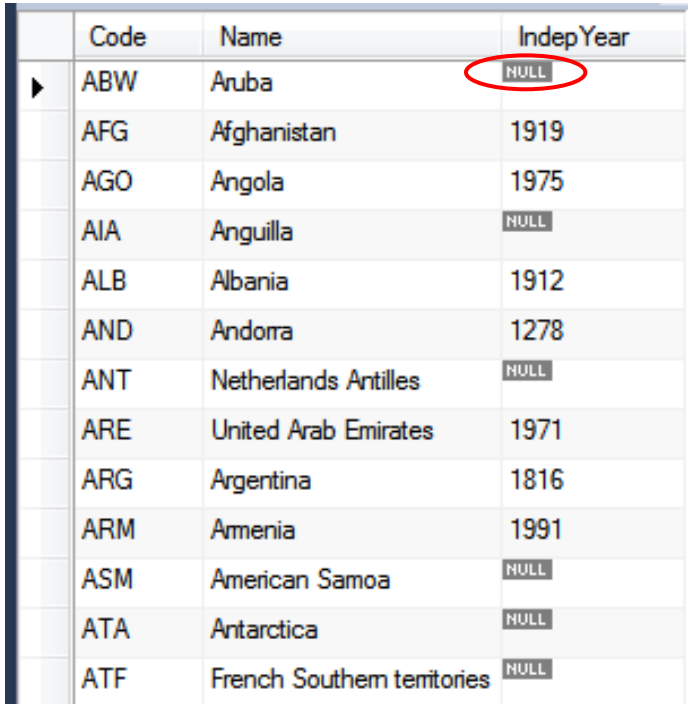
- MySQL contains several datatypes. Some of the most common ones are: char, varchar, int, float, date and enum.

- A full listing (MySQL and other SQL databases) can be seen at:
  - SQL datatypes @ w3schools.com.

# The value NULL

- NULL represents a cell that doesn't contain any data.

- NOTE:
  - NULL is not the same as the value 0.
  - NULL is not the same as a blank / space.
  - NULL is simply "nothing".

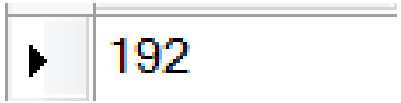


| Code | Name | IndepYear |
|---|---|---|
| ABW | Aruba | NULL |
| AFG | Afghanistan | 1919 |
| AGO | Angola | 1975 |
| AIA | Anguilla | NULL |
| ALB | Albania | 1912 |
| AND | Andorra | 1278 |
| ANT | Netherlands Antilles | NULL |
| ARE | United Arab Emirates | 1971 |
| ARG | Argentina | 1816 |
| ARM | Armenia | 1991 |
| ASM | American Samoa | NULL |
| ATA | Antarctica | NULL |
| ATF | French Southern territories | NULL |

# NULL can trick us

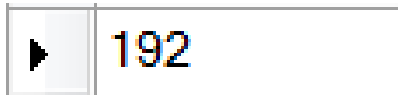
```sql
SELECT COUNT(*) AS NumberOfContries
FROM country;
```


```sql
SELECT COUNT(*) AS NumberOfContries
FROM country
WHERE IndepYear > 1814 OR IndepYear <= 1814;
```

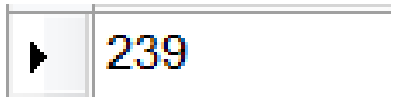


# NULL can trick us – cont.

```
SELECT COUNT(*) AS NumberOfContries
FROM country
WHERE IndepYear > 1814
OR IndepYear <= 1814
OR IndepYear = NULL;
```




```
SELECT COUNT(*) AS NumberOfContries
FROM country
WHERE IndepYear > 1814
OR IndepYear <= 1814
OR IndepYear IS NULL;
```



# SQL operators

- Operators in general:

| | |
|---|---|
| `=` | equal to (*not* wildcards!) |
| `<>` *or* `!=` | different from |
| `<` | less than |
| `>` | greater than |
| `<=` | less than or equal to |
| `>=` | greater than or equal to |

| | |
|---|---|
| `like` | equal to (wildcards ok) |
| `in` | within a set |
| `between` | between values a `and` b |
| `_` | wildcard, one character |
| `%` | wildcard, several characters |
| `is null` | or opposite: `is not null` |

- Logical operators:

| | |
|---|---|
| `and` | (← as operator says) |
| `or` | (← as operator says) |
| `not` | (← as operator says) |

# DISTINCT

- Sometimes we get duplicate results in a query.
  - Typically when we only select a few columns out of a larger total.

- To skip any duplicate results from a select, use DISTINCT:

```
SELECT DISTINCT CountryCode
FROM city
ORDER BY CountryCode ASC
```

# GROUP BY and HAVING

- `GROUP BY` will let us split group-results into more than one row.

- We get group-results when we use functions like:
  - `COUNT`, `SUM`, `AVG`, ...

- If we additionally want to exclude rows out of the *result*:
  - We can't use ~~`WHERE`~~, we need to use `HAVING`.

# GROUP BY and HAVING – cont.

- `WHERE` excludes rows *before* grouping.
- `HAVING` excludes rows *after* grouping.
- SQL performs its different steps in this order:
  - `FROM`
  - `WHERE`
  - `GROUP BY`
  - `HAVING`
  - `SELECT`
  - `ORDER BY`

(Thus, *not* in the written SQL statement order, where SELECT is first.)

# GROUP BY and HAVING – cont.

```
SELECT COUNT(*), MIN(SurfaceArea), MAX(IndepYear),
AVG(LifeExpectancy), SUM(GNP)
FROM country;
```

```
SELECT Continent, COUNT(*), MIN(SurfaceArea),
MAX(IndepYear), AVG(LifeExpectancy), SUM(GNP)
FROM country
GROUP BY Continent;
```

```
SELECT Continent, COUNT(*), MIN(SurfaceArea),
MAX(IndepYear), AVG(LifeExpectancy), SUM(GNP)
FROM country
GROUP BY Continent
HAVING COUNT(*) > 20 AND MIN(SurfaceArea) > 20;
```

# GROUP BY and HAVING – cont.

```
SELECT Continent, COUNT(*), MIN(SurfaceArea),
MAX(IndepYear), AVG(LifeExpectancy), SUM(GNP)
FROM country
WHERE IndepYear < 1950
GROUP BY Continent
HAVING COUNT(*) > 20 AND MIN(SurfaceArea) > 20;
```

- The difference between this and the previous is: "WHERE IndepYear < 1950".

- What's the difference in the result?
  - We lost North-America and Africa, because they no longer fulfilled the COUNT criteria.
  - We gained Asia, because we removed the rows hindering it from fulfilling the MIN criteria.
  - WHERE removes rows *before grouping*, which then can change the results of `HAVING`.

# SQL – SELECT queries

- For what we have learned so far, the full syntax order is:

```
 SELECT    column* [AS name]
 FROM      table
[WHERE     clause]
[GROUP BY  column*]
[HAVING    clause]
[ORDER BY  column*]
```


** or SQL function: count, sum, …*

# Today's exercises & looking ahead

- Now: 2 hours of exercises.

- Exercises are found on Canvas. Short summary:
  - More training on SELECT statements (queries), incorporating what we just learned (like GROUP BY and HAVING).

- Main contents for the next lesson:
  - More on the concept of keys in relational databases.
  - A different type of SQL statements: Changing the data contents.

The