13/07/2019

HackYourFuture Databases - Lesson 2

Class 4

Borja Romero
Lessons 1,2,3

Gina Stavropoulou
Lessons 1,2

Geert Van Pamel
Lessons 2,3

Who is who?

Geert Van Pamel

- Belgian, Dutch speaking + EN, FR, DE
- Engineer
- Worked in the IT industry from 1984-2018
- Ran multiple LAMP (Perl, PHP) servers
- Volunteer for Wikimedia; active on Wikipedia, Commons, and Wikidata
- Also active on OpenStreetMap

Photo: Ronn, CC BY-SA 4.0

Revision (Kahoot!)

Let’s play:

1. Insert the quiz number shown on the screen
2. Read the question
3. You have 20’ to choose one of the answers
4. Answer the questions faster than the others
5. Multiple answers can possibly be right
6. Some answers are tricky (think twice)

First things first

Follow the link, download and source imdb2.sql. (We have updated it for the purpose of this lesson):

https://drive.google.com/open?id=1qgG9lJdrG-_SxSUXUoWkBO-Lvb10cqMh

Source /directory/to/folder/imdb2.sql;

(it will substitute the previous imdb).

Revision (SELECT Text)

- The text is always surrounded by quotes.
- % represents any combination of characters in any amount (including 0) but only in **LIKE** conditions.
  - What happens if you use a % in a where name **=** ‘%Brussels%’; ?
  -
- How do you include a real % in the query?
  - textPercentage LIKE ‘100\%’ -- \ is what we call a Escape character.
- Can’t we use “=” ? What is the difference between:
  - where name = ‘Brussels’   AND   WHERE name LIKE ‘Brussels’

Basically none, although in the IT world you always get surprises:

```
INSERT INTO t1 VALUES ('davyjones');
SELECT * FROM t1 WHERE c10 = 'davyjones'; -- yields 1 row
SELECT * FROM t1 WHERE c10 LIKE 'davyjones'; -- yields 0 rows
```

Revision (ORDER BY)

The data order of a SELECT is generally undefined.

Why:

- Mathematically in a SET there is no order defined
- Technically, based on the optimiser decisions, there can be an order (from the primary key)
- If you do a GROUP BY, the output is normally sorted

To choose the sort order, you better use ORDER BY explicitly

REVISION (COUNTING)

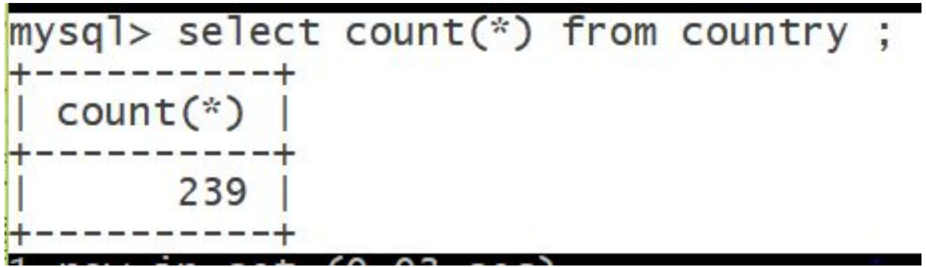
`Count(*) FROM Country` <- 239

Vs

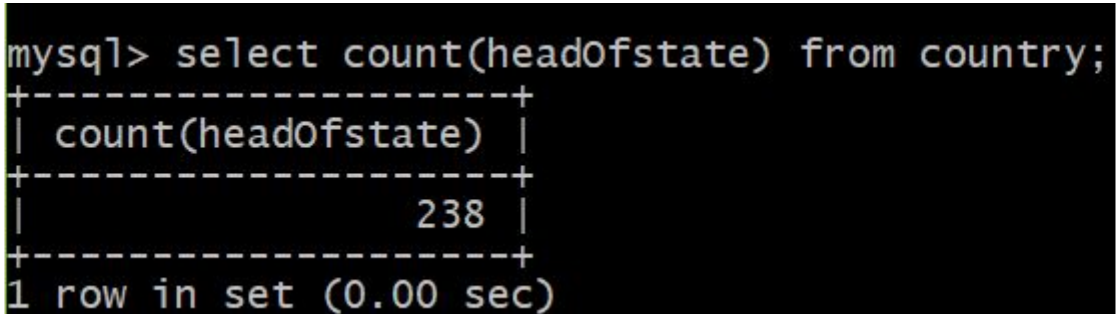
`Count(1) FROM Country` <- 239

Vs

`Count(headofstate) FROM Country;` <- 238 !!! **What?**



```
mysql> select count(*) from country ;
+----------+
| count(*) |
+----------+
|      239 |
+----------+
```



```
mysql> select count(headOfstate) from country;
+--------------------+
| count(headOfstate) |
+--------------------+
|                238 |
+--------------------+
1 row in set (0.00 sec)
```

-> Only NON-NULL values are counted!!

Why there is a difference?

```
select count(*) from country where HeadOfState is null;
```

```
1
```

```
select case when HeadOfState is null then 'NULL' else
'VALUE' end Data, count(*) Count from country group by Data;
```

```
| Data  | Count |
+-------+-------+
| NULL  |     1 |
| VALUE |   238 |
```

REVISION (CREATE)

Create a new database in your system: `CREATE DATABASE db_name;`

Create a new table:

```
CREATE TABLE customers
(    customer_id INT(5),
     FirstName VARCHAR(50) NOT NULL,
     LastName VARCHAR(50) NOT NULL,
     age INT (3), preferences TEXT,
     PRIMARY KEY (customer_id));
```

Always-> **Column Name + Data type**, Optional -> **Constraints:**

The `NOT NULL` indicates that the inserted value cannot be `NULL`.
The `DEFAULT value` is used to specify the default value of the column.
The `PRIMARY KEY` specifies that values are the unique identifiers.

REVISION (INSERT)

```
INSERT INTO STUDENT VALUES (1, ‘Rob’, ‘Smith’, 54, ‘Course A’);
```

But also:

```
INSERT INTO STUDENT (Id, FirstName, LastName, Age, CourseTitle)
VALUES (1, ‘Rob’ , ‘Smith’, 54, ‘Course A’);
```

Or

```
INSERT INTO STUDENT (Id, FirstName, LastName, Age)   -- No course title => NULL value
VALUES (1, ‘Rob’ , ‘Smith’, 54);
```

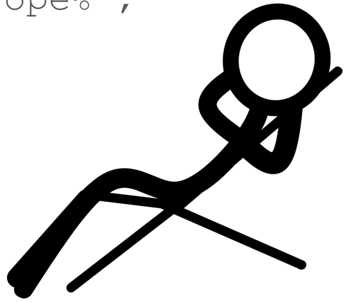
What do you think that will happen to the Course Title?

REVISION (INSERT)

But even more:

Imagine you need to copy all cities from Europe to a new table.

```
INSERT INTO EuropeanCities (name, population, region)
SELECT name, population, region FROM city WHERE region LIKE ‘%Europe%’;
```

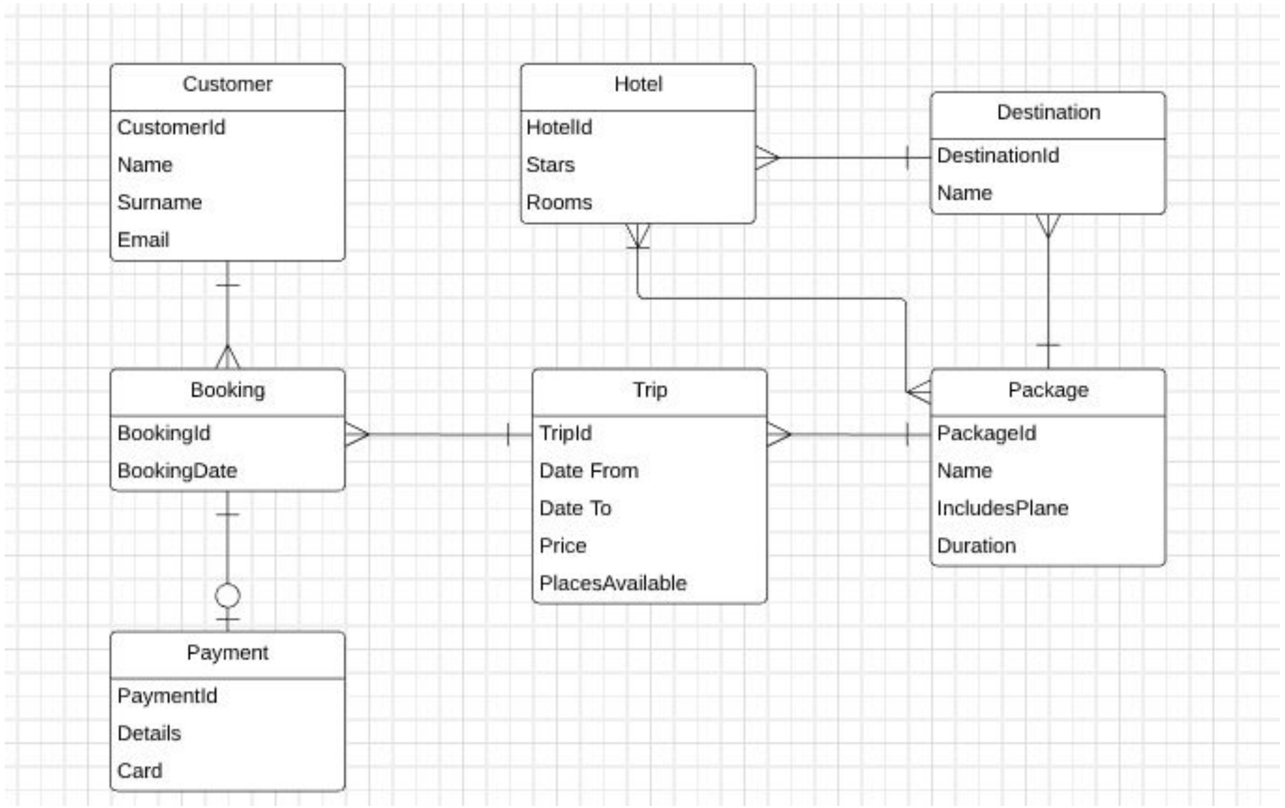


This above is combining the INSERT with the SELECT.

You could even have created the table:

```
CREATE TABLE EuropeanCities
SELECT name, population, region FROM city WHERE region LIKE ‘%Europe%’;
```

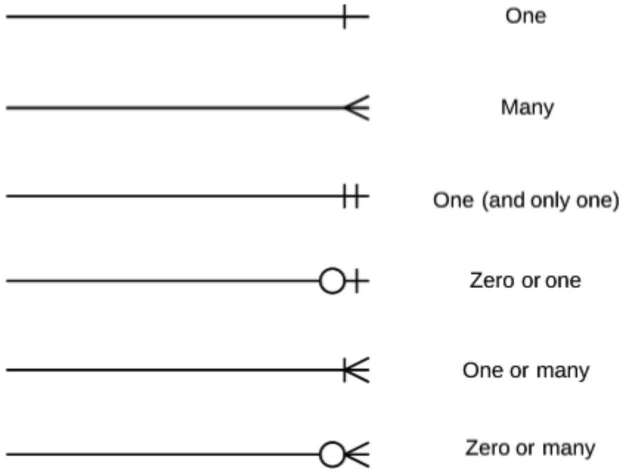
REVISION (ERD)

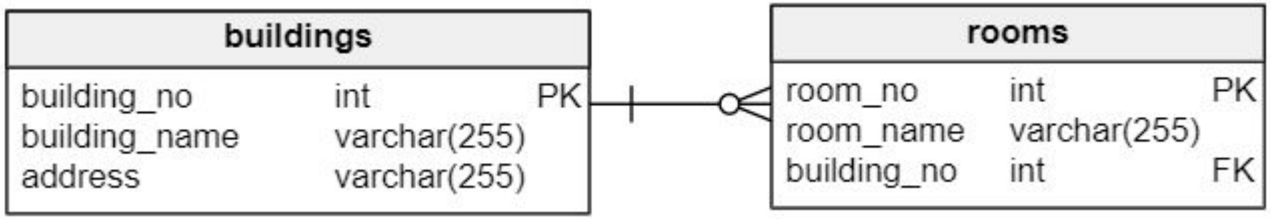


Cardinality & Modality

Cardinality : the type of relationship between rows of tables. 3 cases: One-to-one, one-to-many, many-to-many

Modality: if the relationship between the entities is mandatory (NOT NULL, “1”) or not (NULL, “0”).



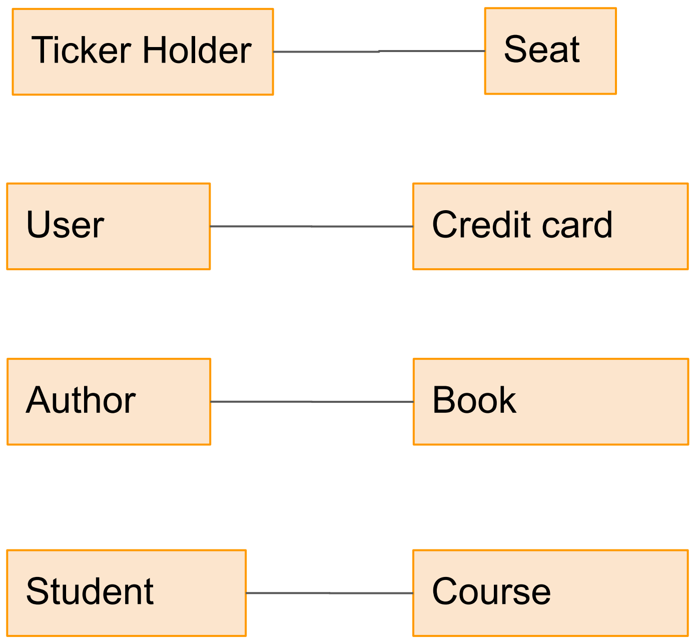


buildings		
building_no	int	PK
building_name	varchar(255)	
address	varchar(255)	

rooms		
room_no	int	PK
room_name	varchar(255)	
building_no	int	FK

Cardinality & Modality

What is the relationship between the following entities?



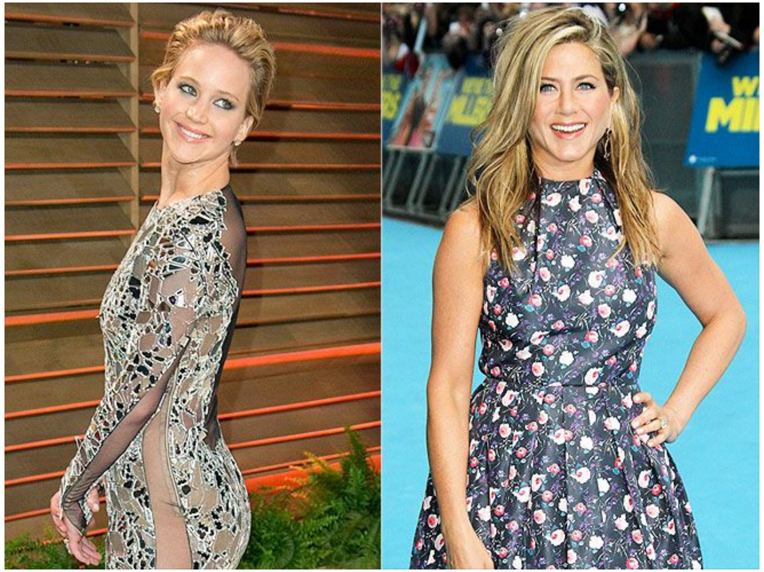
https://www.lucidchart.com

DISTINCT

What are the names of the actresses that we have in the database?

```
SELECT fname FROM actors WHERE gender = 'f';
```

```
+----------+
| fname    |
+----------+
| Jennifer |
| Anne     |
| Meryl    |
| Jennifer |
| Helena   |
+----------+
5 rows in set (0.00 sec)
```



DISTINCT (2)

Great but I just wanted to see the names, I have seen 2 Jennifers, any way to have only the set of different names without duplicates? Let's try!

```
mysql> SELECT DISTINCT fname FROM actors WHERE gender ='f';
+----------+
| fname    |
+----------+
| Jennifer |
| Anne     |
| Meryl    |
| Helena   |
+----------+
4 rows in set (0.00 sec)
```

(morph created between Jennifer Lawrence and Jennifer Anniston @ www.morphThing.com)

DISTINCT (Exercise time!)

Two easy ones:

1. Connect to imdb (the new one).
2. Make a list with all the directors.

```
SELECT DISTINCT director
FROM films;
```

3. Connect to world.
4. Make a list of all the regions where there is a country that starts with ‘Q’.

```
SELECT DISTINCT region
FROM country WHERE name
LIKE "Q%";
```

DISTINCT (Exercise time!)

A difficult one:

1. Connect to the world database
2. Use the countryLanguage table.
3. Tell us the languages that are official in a country but they have a percentage of 0%. O_o

```
SELECT DISTINCT language
FROM countryLanguage
WHERE isOfficial = 'T' AND
percentage = 0;
```

STRING functions

Try the following:

```
SELECT fname, lname FROM actors WHERE gender ='f';

SELECT CONCAT ( fname, ' ', lname) FROM actors WHERE gender ='f';

SELECT CONCAT ( fname, ' ', UPPER (lname))
FROM actors WHERE gender ='f';

SELECT CONCAT ( substring(fname,1,1), '. ', UPPER (lname)) FROM
actors WHERE gender ='f';
```

Revision (SELECT FUNCTIONS)

COUNT, MAX, MIN, (SUM & AVG).

Use world.sql

- How many countries belong to the region “Northern Africa”?
- How many countries belong to the region “Southeast Asia”?

- What is the population of the most populated city in Spain?
- What is the population of the most populated city in Belgium?
- Do the same for the least populated (in China or Japan)

Revision (SELECT FUNCTIONS)

COUNT, MAX, MIN.

Use world.sql

- How many countries belong to the region “Northern Africa”?

```
SELECT COUNT(*) FROM country WHERE region = "Northern Africa";
```

- How many countries belong to the region “Southeast Asia”?

```
SELECT COUNT(*) FROM country WHERE region = "Southeast Asia";
```

Revision (SELECT FUNCTIONS)

- What is the population of the most populated city in Spain?

```
select MAX(population) FROM city WHERE countrycode = "ESP";
```

- What is the population of the most populated city in Belgium?

```
SELECT MAX(population) FROM city WHERE countrycode = "BEL";
```

- Do the same for the least populated (in China or Japan):

```
SELECT MIN(population) FROM city WHERE countrycode = "JPN";
```

GROUP BY

➔ Pleaaaase, stop asking for regions! But what if …

```
SELECT COUNT(*), region
FROM country
GROUP BY  region;

SELECT MAX(population), countryCode
FROM city
GROUP BY countryCode;
```

…. gave the same results without asking many times.

GROUP BY (Exercise time!)

Now you!

Connect/use to the imdb database

1. Find how many actors and actresses are older than 60 grouped by their gender.
2. Find what is the maximum age for actors and actresses with only one query.

Connect/use the world database

3. Find in the country table, what is the highest life expectancy per region in regions containing the string Europe (‘%Europe%’)

GROUP BY (Exercise time!)

Find how many actors and actresses are older than 60 grouped by their gender.

```
SELECT gender, COUNT(*) FROM actors WHERE age>60 GROUP BY gender;
```

Find what is the maximum age for actors and actresses with only one query.

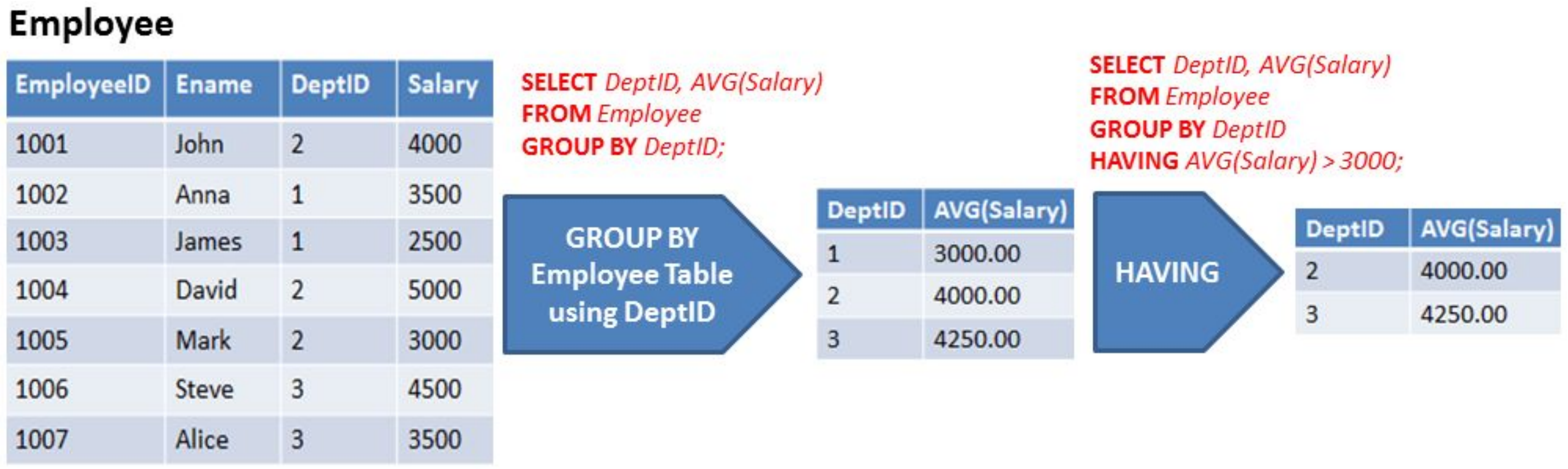
```
SELECT gender, MAX(age) AS max_age FROM actors GROUP BY gender;
```

Find in the country table, what is the highest life expectancy per region in regions containing the string Europe (‘%Europe%’)

```
SELECT region, MAX(LifeExpectancy) AS max_LIFEEXP FROM country GROUP
BY region HAVING region LIKE '%Europe%';
```

HAVING

A way to filter the aggregated (grouped) results (not the rows)



Employee

EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1002	Anna	1	3500
1003	James	1	2500
1004	David	2	5000
1005	Mark	2	3000
1006	Steve	3	4500
1007	Alice	3	3500

```
SELECT DeptID, AVG(Salary)
FROM Employee
GROUP BY DeptID;
```

DeptID	AVG(Salary)
1	3000.00
2	4000.00
3	4250.00

```
SELECT DeptID, AVG(Salary)
FROM Employee
GROUP BY DeptID
HAVING AVG(Salary) > 3000;
```

DeptID	AVG(Salary)
2	4000.00
3	4250.00

HAVING (is not WHERE)

List the country codes of the countries having at least 10 cities with 1 million inhabitants:

```
SELECT countrycode, COUNT(*) AS cityCount FROM city WHERE population
>= 1000000 GROUP BY countrycode HAVING cityCount >= 10;
```

HAVING is applied after the **GROUP BY** phase whereas **WHERE** is applied before the **GROUP BY** phase.

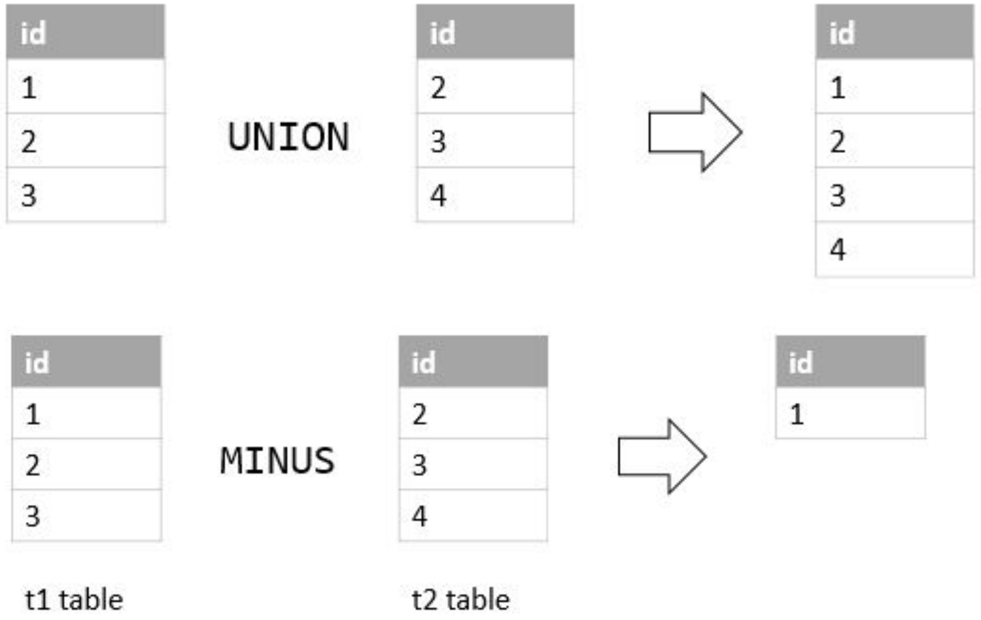
~~`SELECT a, COUNT(*) AS c`~~
~~`FROM mytable GROUP BY a`~~
~~`WHERE c > 1;`~~

→

```
SELECT a, COUNT(*) AS c
FROM mytable GROUP BY a
HAVING c > 1;
```

UNION and MINUS

What if I want to access data that is in two tables together in one?



id
1
2
3

UNION

id
2
3
4

id
1
2
3
4

id
1
2
3

MINUS

id
2
3
4

id
1

BREAK!

Revision(PRIMARY KEY)

A field that uniquely identifies a record in the table.

ID	Name	Last Name	email
1	Walder	Frey	walder19@gmail.com
2	Walder	Frey	wfrey@gmail.com
3	Walder	Frey	walderfrey@gmail.com
4	Walder	Frey	wfrey91@gmail.com

Revision(PRIMARY KEY)

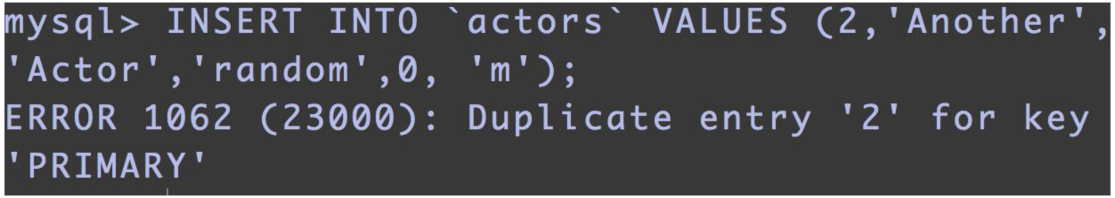
show create table actors;

```
CREATE TABLE actors (
      aid int(11),
      fname varchar(50),
      lname varchar(50),
      biography text,
      age tinyint(2),
      gender varchar(1),
       PRIMARY KEY (aid)
)
```

OR

```
CREATE TABLE actors (
      aid int(11) PRIMARY KEY,
      fname varchar(50),
      lname varchar(50),
      biography text,
      age tinyint(2),
      gender varchar(1)
)
```

If you try to insert a new record (actor) with an id value that is already in the db you will get an error:

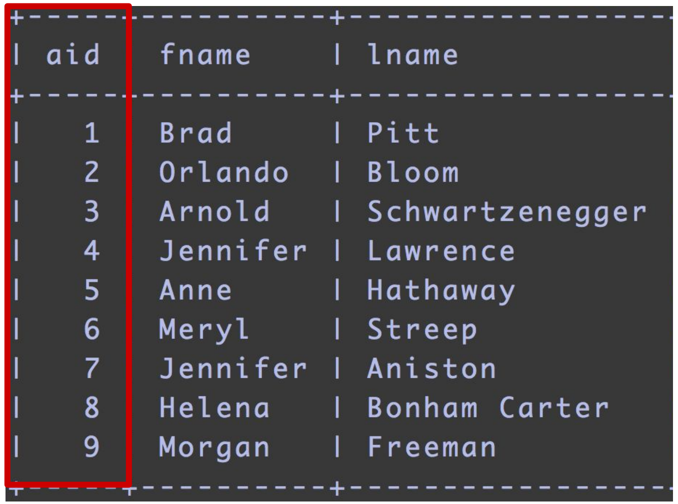


```
mysql> INSERT INTO `actors` VALUES (2,'Another',
'Actor','random',0, 'm');
ERROR 1062 (23000): Duplicate entry '2' for key
'PRIMARY'
```

FOREIGN KEY

When you refer the primary key column of a table in another table then it becomes a **FOREIGN KEY.** It is a column that uniquely identifies rows in another table.

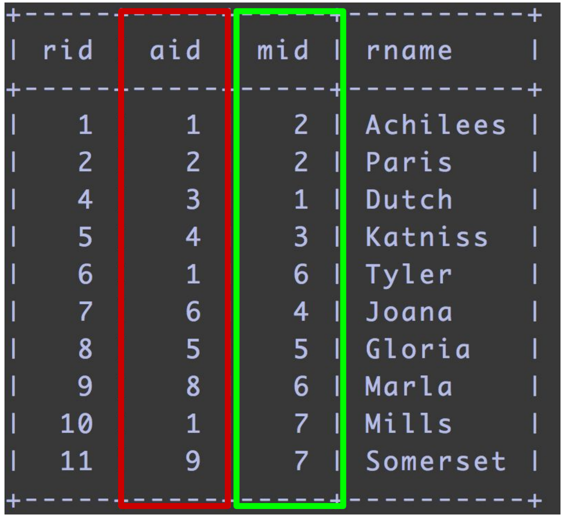
PK



```
+-----+-----------+--------------------
| aid | fname     | lname
+-----+-----------+--------------------
|   1 | Brad      | Pitt
|   2 | Orlando   | Bloom
|   3 | Arnold    | Schwartzenegger
|   4 | Jennifer  | Lawrence
|   5 | Anne      | Hathaway
|   6 | Meryl     | Streep
|   7 | Jennifer  | Aniston
|   8 | Helena    | Bonham Carter
|   9 | Morgan    | Freeman
+-----+-----------+--------------------
```

Actors table

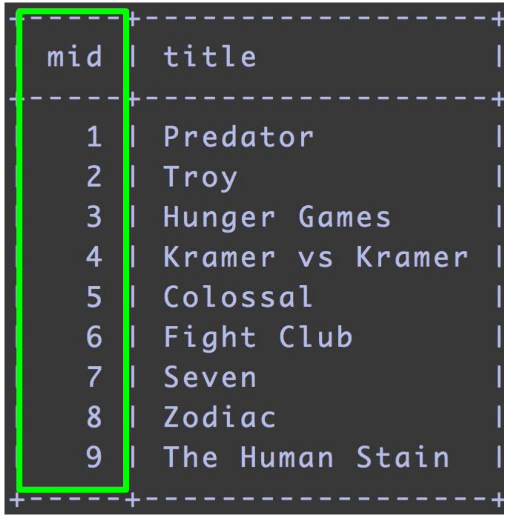
PK FK FK



```
+-----+-----+-----+----------+
| rid | aid | mid | rname    |
+-----+-----+-----+----------+
|   1 |   1 |   2 | Achilees |
|   2 |   2 |   2 | Paris    |
|   4 |   3 |   1 | Dutch    |
|   5 |   4 |   3 | Katniss  |
|   6 |   1 |   6 | Tyler    |
|   7 |   6 |   4 | Joana    |
|   8 |   5 |   5 | Gloria   |
|   9 |   8 |   6 | Marla    |
|  10 |   1 |   7 | Mills    |
|  11 |   9 |   7 | Somerset |
+-----+-----+-----+----------+
```

Roles table

PK



```
+-----+-------------------+
| mid | title             |
+-----+-------------------+
|   1 | Predator          |
|   2 | Troy              |
|   3 | Hunger Games      |
|   4 | Kramer vs Kramer  |
|   5 | Colossal          |
|   6 | Fight Club        |
|   7 | Seven             |
|   8 | Zodiac            |
|   9 | The Human Stain   |
+-----+-------------------+
```

Films table

See FOREIGN KEY definition

show create table roles;

```
CREATE TABLE roles (
  aid int(11) NOT NULL,
  mid int(11) NOT NULL,
  rname varchar(50) DEFAULT NULL,
  rid int(11) NOT NULL,
  PRIMARY KEY (rid),
  KEY mid (mid),
  KEY aid (aid),
  CONSTRAINT roles_ibfk_1 FOREIGN KEY (mid) REFERENCES films
(mid),
  CONSTRAINT roles_ibfk_2 FOREIGN KEY (aid) REFERENCES actors
(aid)
) |
```

FOREIGN KEY Constraints

Before you can insert a role for an actor (foreign key), you need to insert the actor in the actors table (primary key):

- You cannot add a role for an actor if the actor does not exist:

  TRY: `INSERT INTO roles (aid, mid, rname, rid) VALUES ( 32, 2,'John',42);`

- You cannot remove an actor if he still has active roles:

  TRY: `DELETE FROM actors WHERE aid = 1;`

- You cannot delete the table actors while you have a foreign key constraint:

  TRY: `DROP TABLE actors;`

Aliases

Help us make the queries shorter and more readable. Defined with the keyword **AS**:

```
SELECT actors.fname, roles.rname FROM actors, roles WHERE actors.aid =roles.aid;
```

⇩

```
SELECT a.fname, r.rname FROM actors AS a, roles  AS r WHERE a.aid =r.aid;
```

JOIN - Time to start navigating!

Why are we using JOINS?

Database is structured in (simple) tables to avoid:

- data duplication (storage)
- repeated columns (think about changing business rules)
- update anomalies (multiple updates instead of single row update)
- delete anomaly (loss of “reference” data when last occurence is deleted)

Think about (logical) data model and normalisation.

Tables contain columns (PRIMARY KEY <-- FOREIGN KEY).

JOINS are used to combine tables using “related” columns.

JOIN -

```
mysql> SELECT code, name, region FROM  country WHERE code = 'USA';
+-------+-----------------+------------------+
| Code | name             | region           |
+-------+-----------------+------------------+
| USA  | United States | North America |
+-------+-----------------+------------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM city WHERE countryCode = 'USA' LIMIT 1;
+------+-------------+-----------------+------------+-------------+
| ID   | Name        | CountryCode | District     | Population |
+------+-------------+-----------------+------------+-------------+
| 3793 | New York | USA               | New York |    8008278 |
+------+-------------+-----------------+------------+-------------+
1 row in set (0.22 sec)
```

Typical uses for a JOIN

- Join a reference table (get the actor’s name based on his actor’s id)
- Get 1:M relationships
    - get the children from parents (like a city is a *child* of a country)
- Get the transaction history:
    - get all orders from a customer
    - get the list of products and their quantity in an order
    - get the product price (start date, end date)
    - get all invoice items from an invoice
    - get the list of payments for a customer
    - get unpaid invoices (left outer join with payments)
- Reporting
    - Star diagram (customer name, customer segment, product name, product category, region)

JOINS

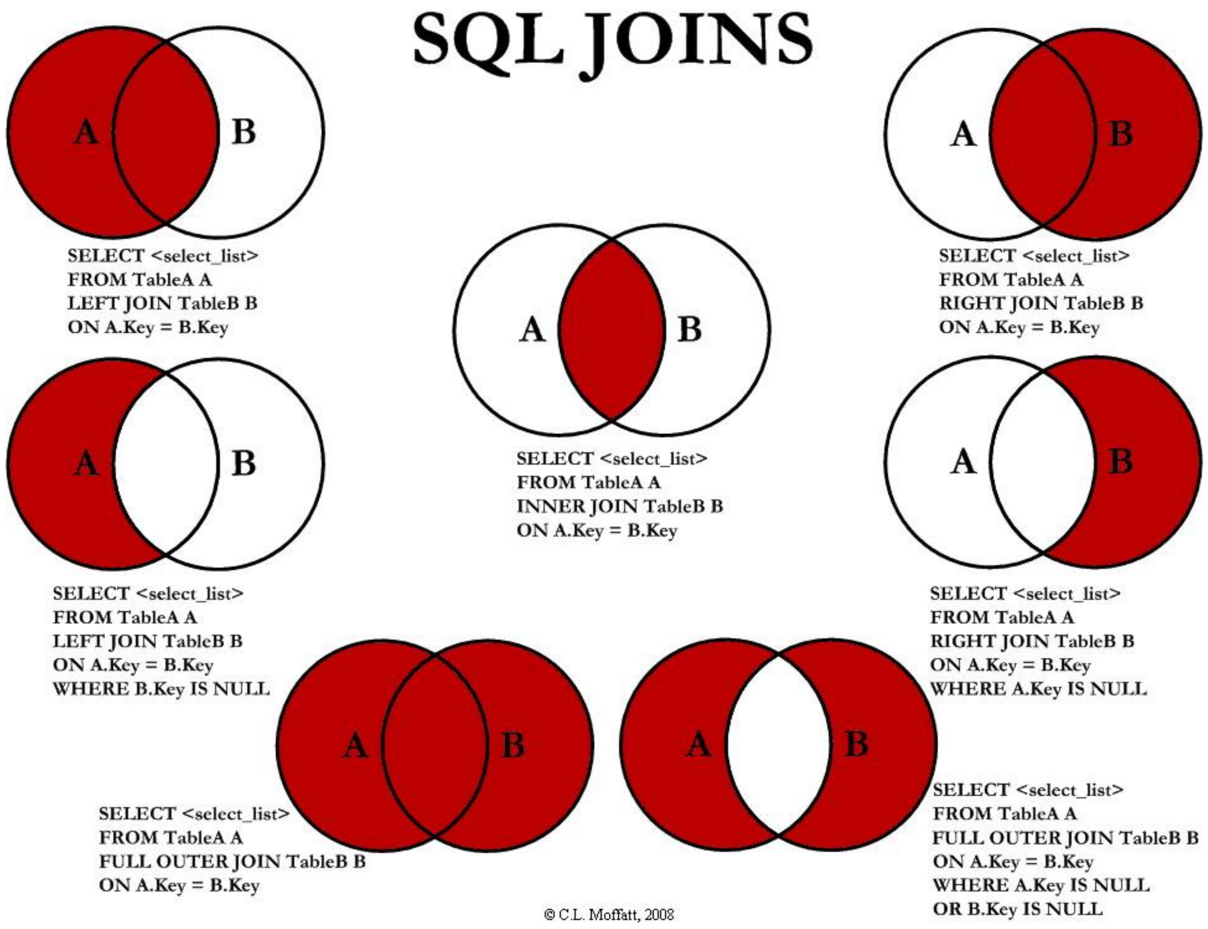
Get all the countries where English is the official language.

With a WHERE statement:

```
SELECT Name FROM country, countrylanguage WHERE Countrycode = Code
AND Language = 'English';
```

This corresponds to an INNER JOIN.

JOIN



```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

INNER JOIN

Get all the countries where English is the official language.

With an INNER JOIN statement:

```
SELECT Name FROM country INNER JOIN countrylanguage ON Countrycode =
Code WHERE Language = 'English';
```

```
SELECT Name FROM country INNER JOIN countrylanguage ON Countrycode =
Code AND Language = 'English';
```

You can omit the word INNER.

OUTER JOIN

Get all countries which does not have an (official) language.

```
SELECT Name FROM country LEFT OUTER JOIN countrylanguage ON
Countrycode = Code WHERE CountryCode IS NULL;
```

An outer join is typically used when there is no data available.

Otherwise an inner join would not provide results.

Depending on the order of the tables you use LEFT of RIGHT join.

You can omit the word OUTER.

subSELECT = single value SELECT between ( )

Find the actors who have played 2 roles:

```
SELECT fname, lname FROM actors WHERE aid = (SELECT aid FROM roles
GROUP BY aid HAVING COUNT(*) = 2);
```

```
SELECT fname, lname FROM actors AS a JOIN roles AS r ON a.aid = r.aid
GROUP BY r.aid HAVING COUNT(*) = 2;
```

```
SELECT fname, lname, rname, title FROM actors AS a JOIN roles AS r ON
a.aid = r.aid JOIN films AS f ON f.mid = r.mid WHERE r.aid = (SELECT
aid FROM roles GROUP BY aid HAVING COUNT(*) = 2);
```

Many to many relationship M:M

How would you implement: M:M relationships are split in 2 x 1:M relationships.

Person --< TeamMembership >-- Team

- Give the details of possible attributes, primary keys, foreign keys
- What date columns would you add?
- … think about describing the history ...
- What constraints? (dates!)
- What other tables could you add? (skills, roles, training, projects, application, systems, server, buildings, location)

Very much used in real world applications.

Data quality

- JOIN is normally performed between PRIMARY KEY and FOREIGN KEY
- Pay attention not to create false data when joining (e.g. forget one AND clause)
- Pay attention to priority NOT AND OR -> use ( ) when necessary

Find the actors and their roles for all the male actors or the actors with age above 50: ( ) required

```
SELECT fname, lname, rname FROM actors, roles WHERE actors.aid = roles.aid
AND ( gender = 'm' OR age > 50 );
```

DELETE rows

To remove data rows from a table:

```
DELETE FROM table WHERE … ;
```

Reasons:

- Remove old data
- Delete wrong data
- Privacy reasons
  - e.g. when a customer does not order during 1 year we cannot contact him anymore for marketing purposes
    - We must still keep his invoicing and payment data for 10 years
    - Ordering details can be deleted after e.g. 5 years
  - When an employee does not work for the company any more we need to delete his national number
    - We need to keep the pay slip for 10 years

See GDPR (European Data protection rules - new EU privacy law as of 2018)

Data privacy rules

Clear obsolete data

```
UPDATE employee
SET national_number = NULL
WHERE employee_id = 1003;
```

```
UPDATE employee
SET national_number = NULL
WHERE employee_revoke_dt < date - 365;
```

ALTER TABLE

ALTER TABLE can be used to change the structure of a table.

- Add a column
- Drop a column
- Increase the size of a string column
- Change the data type of a column
- Add or drop keys (primary, foreign, lookup)

ALTER TABLE - Add a column

Imagine you have a table `alterex` containing orders.

The customer would like to have a delayed order. So you need a delivery date.

```
ALTER TABLE alterex ADD deliveryDt DATE;

DESC alterex;
```

```
+------------+---------+------+-----+---------+-------+
| Field      | Type    | Null | Key | Default | Extra |
+------------+---------+------+-----+---------+-------+
| ordernum   | char(9) | YES  |     | NULL    |       |
| deliveryDt | date    | YES  |     | NULL    |       |
+------------+---------+------+-----+---------+-------+
```

ALTER TABLE - Drop a column

Dropping a column means that you also delete the data.

```
ALTER TABLE alterex DROP deliveryDt;

DESC alterex;

+------------+---------+------+-----+---------+-------+
| Field      | Type    | Null | Key | Default | Extra |
+------------+---------+------+-----+---------+-------+
| ordernum   | char(9) | YES  |     | NULL    |       |
+------------+---------+------+-----+---------+-------+
```

ALTER TABLE - Modify data type

Modifying the data type of a column from one type to another - or just altering the allowed size.

```
ALTER TABLE alterex MODIFY ordernum INT(9);

DESC alterex;
```

```
+----------+--------+------+-----+---------+-------+
| Field    | Type   | Null | Key | Default | Extra |
+----------+--------+------+-----+---------+-------+
| ordernum | int(9) | YES  |     | NULL    |       |
+----------+--------+------+-----+---------+-------+
```

ALTER TABLE - Add/Drop Primary Key

```
ALTER TABLE alterex ADD [CONSTRAINT PK_name]

 PRIMARY KEY (ordernum);

DESC alterex;

+----------+---------+------+-----+---------+-------+
| Field    | Type    | Null | Key | Default | Extra |
+----------+---------+------+-----+---------+-------+
| ordernum | char(9) | NO   | PRI | NULL    |       |
+----------+---------+------+-----+---------+-------+

ALTER TABLE alterex DROP PRIMARY KEY;
```

ALTER TABLE - Add/Drop FOREIGN Key

```
CREATE table otherex(id INT(3) PRIMARy KEY);

ALTER TABLE alterex ADD other INT(3);

ALTER TABLE alterex ADD CONSTRAINT FK_name FOREIGN KEY
(other) REFERENCES otherex(id);

+----------+---------+------+-----+---------+-------+
| Field    | Type    | Null | Key | Default | Extra |
+----------+---------+------+-----+---------+-------+
| ordernum | char(9) | NO   | PRI | NULL    |       |
| other    | int(3)  | YES  | MUL | NULL    |       |
+----------+---------+------+-----+---------+-------+
ALTER TABLE alterex DROP FOREIGN KEY fk_name;
```

In case of an error you can always source the sql file again

Homework 1 (More Queries)

WORLD:

1. What is the name and the population of the most populated city in India?
2. List the names of the countries having at least 3 cities with 3 million inhabitants.
3. What is the number of all the official languages? List the country name and their official languages.
4. Find all the countries that have only one official language
5. Find which countries do not have a capital.
6. Which country has the lowest population?
7. Make a list with all the languages spoken in Eastern Africa.

IMDB

8. Find the minimum and the maximum age of the actors per gender.
9. Find how many actors are in their 20's, 30's, 40's, 50's etc (grouped by decade).
10. Add a column to the films table for storing the duration (runtime) or each film.
11. Alter the data type of column age to INT.
12. Print the names and biographies of the actors in this format "ANNE HATHAWAY BIO: 1 golden globe"
13. Delete the column biography from films.

Homework 2

1. Modify your ERD after what you saw today.
2. Create the a database, create the relevant tables and populate them with trips, customers, payments and a few more (at least 10 in each table).
3. Use Primary and Foreign Keys to link the tables.
4. Create the queries (using JOINS) to tell us:
    a. How many trips did a customer book.
    b. What are the destinations of your favourite package?
    c. Which are the packages that include Paris?
    d. Or any other queries that you can think of (be creative!)

Homework 3 (optional)

Create a new ERD based on the imdb database.

Include more entities like: Directors, Oscars etc

Pay attention to Cardinality & Modality

Be creative :)