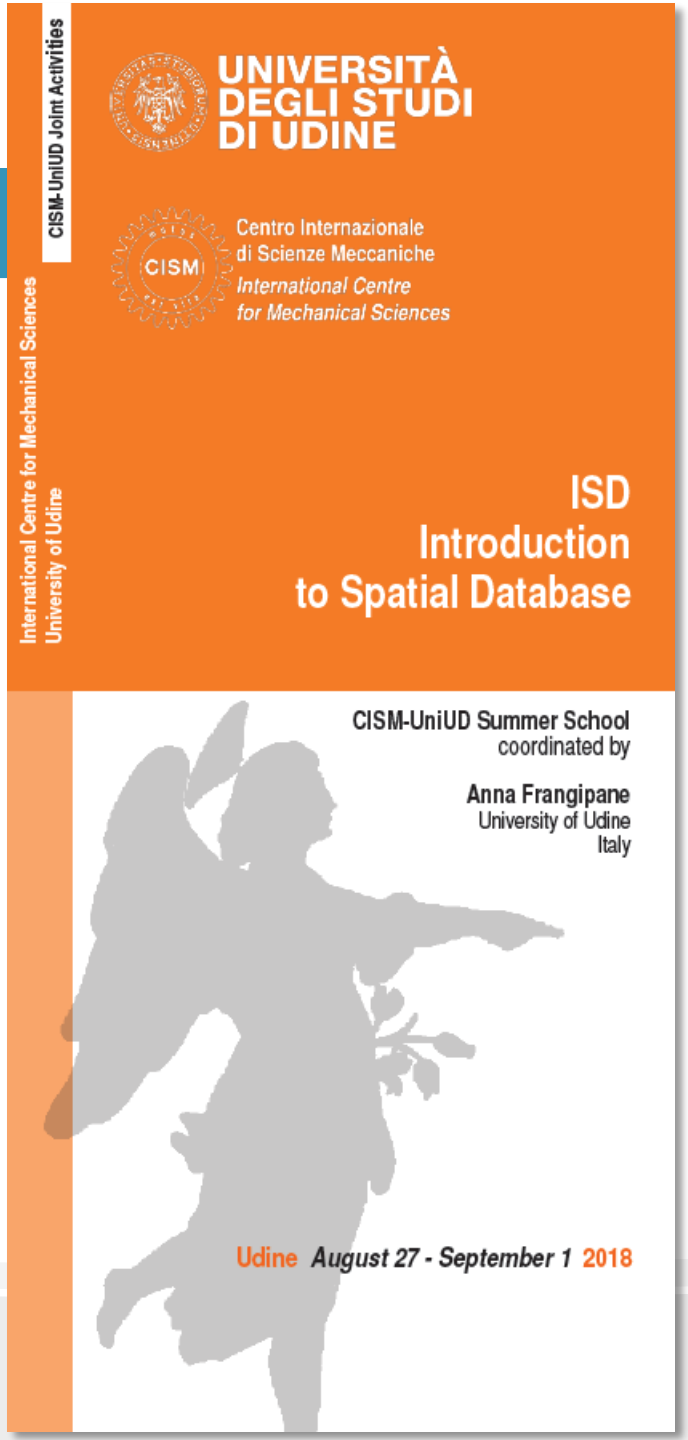

SQL: Join

Marco BASALDELLA, Ph.D.

Department of Mathematics, Computer Science and Physics
University of Udine
marco.basaldella@uniud.it | www.basaldella.it

Outline

1. A brief recap of SQL
2. The **`JOIN`** instruction
3. Exercises

All the material is available online:

`https://github.com/basaldella/isd2018`

MySQL Workbench/Recap

We will use MySQL Workbench as our tool of choice for working with MySQL.

You will work with a database installed in the Artificial Intelligence Laboratory servers. The credentials are:

- hostname: `db.ailab.uniud.it`
- username: `isdXX`, where XX is the number of your PC
- password: `ISD2018`

Table creation

```
CREATE TABLE Table_Name (
    Field_1 DATATYPE FIELD_ATTRIBUTES,
    Field_2 DATATYPE FIELD_ATTRIBUTES,
    ...
    TABLE_ATTRIBUTES
);
```

Insert data/2

```
INSERT INTO table_name (field_1,field_2,...)
VALUES (val_1,val_2,...);
```

Example:

```
INSERT INTO Students (LastName,FirstName) VALUES
("Doe", "John");
```

Result:

ID	LastName	FirstName	Address	Country
1	Doe	John	NULL	NULL

Insert data/3

```
INSERT INTO table_name
VALUES (val_1,val_2,...);
```

Example:

```
INSERT INTO Students
VALUES (100,“Other”, “Anne”, “New York”, “USA”);
```

Result:

	ID	LastName	FirstName	Address	Country
▶	1	Doe	John	NULL	NULL
	100	Other	Anne	New York	USA

Update data/1

We use the **UPDATE** instruction to modify the rows of a table.

- To select the rows we want to update, we use the **WHERE** clause – much like in a **SELECT**
- Syntax:

```
UPDATE table_name
SET field_1=val_1,field_2=val_2,...
WHERE field=val;
```

Delete a Table

The instruction `DROP TABLE` can be used to delete tables.

- Drop a table:
  `DROP TABLE` `table_name`
- Drop the table, but first check if exists:
  `DROP TABLE IF EXISTS` `table_name`

`WARNING`: dropping a table is NOT reversible

Delete Data/1

We use the **DELETE** instruction to delete the rows of a table.

- To select the rows we want to delete, we use the **WHERE** clause – much like in a **SELECT** and exactly as in a **UPDATE**
- Syntax:

```
DELETE FROM table_name
WHERE field=val;
```

Select data

We use the **SELECT** instruction to find information in a table.

- To select the rows that match a certain condition, we use the **WHERE** clause.
- Syntax:

```
SELECT field_1,field_2,...
FROM table_name
WHERE field >= val
GROUP BY / ORDER BY field
LIMIT amount;
```

Select Data/2

`SELECT` can be combined with **lots** of commands:

- **`MIN`**, **`MAX`**, **`SUM`**, **`AVG`**, return the minimum, maximum, the sum, and the average of the values for a given column
- **`ORDER_BY`** orders the results based on one or more attributes
- **`GROUP_BY`** groups the results based on one or more attributes
- **`DISTINCT`** returns only different values

SQL Data types: reference

Type	Name	Description	Example
String	`VARCHAR(length)` `VARCHAR(255)`	Stores a string with length up to the value specified. Usually we set length=255.	`john.doe` `New York`
Numeric	`INT, INTEGER` `FLOAT, DOUBLE, REAL`	Store integers or floating point numbers	`12` `3.1415`
Date	`DATE, TIME` `DATETIME, TIMESTAMP`	Store dates, times, or date *and* time (with precision up to a second).	`01-01-2018` `01-01-2018 00:00:00`
Boolean	`BOOLEAN`	Boolean values	`TRUE, FALSE`

The **`JOIN`** instruction

The **`JOIN`** instruction

- Clause used to combine two (or more) tables
- As in relational algebra: you can think of it as a Cartesian product followed by a **`WHERE`** condition
- Many types of JOIN:
  - **`INNER JOIN`**
  - **`LEFT JOIN`** / **`RIGHT JOIN`**
  - **`CROSS JOIN`**
- Usually we use **`JOIN`** as a synonym for **`INNER JOIN`**

Example: mothers and fathers

Table **MOTHERS**

mother	child
Alice	Bob
Carol	James
Diana	Harry

Table **FATHERS**

father	child
Edward	James
Jim	Bob
John	Judy
Charles	Harry

Step 1: Cartesian Product

Goal: find the parents of the same children

Steps:

- Cartesian product **`Mothers × Fathers`**
  Result: table of size…?

Step 1: Cartesian Product

Goal: find the parents of the same children

Steps:

- Cartesian product **`Mothers × Fathers`**
  Result: table of size
  $(rows_{mother} * rows_{father}, columns_{mother} * columns_{father})$
- Keep only the rows
  `Where` `Mothers.child = Fathers.child`
- Keep only the columns mother, father, children

Cartesian Product

mother	child	father	child
Alice	Bob	Jim	Bob
Carol	James	Jim	Bob
Diana	Harry	Jim	Bob
Alice	Bob	Edward	James
Carol	James	Edward	James
Diana	Harry	Edward	James
Alice	Bob	Charles	Harry
Carol	James	Charles	Harry
Diana	Harry	Charles	Harry
Alice	Bob	John	Judy
Carol	James	John	Judy
Diana	Harry	John	Judy

Keep only the rows we need

mother	child	father	child
Alice	**Bob**	Jim	**Bob**
Carol	James	Jim	Bob
Diana	Harry	Jim	Bob
Alice	Bob	Edward	James
Carol	**James**	Edward	**James**
Diana	Harry	Edward	James
Alice	Bob	Charles	Harry
Carol	James	Charles	Harry
Diana	**Harry**	Charles	**Harry**
Alice	Bob	John	Judy
Carol	James	John	Judy
Diana	Harry	John	Judy

Keep only the rows we need

mother	child	father	child
Alice	Bob	Jim	Bob
Carol	James	Edward	James
Diana	Harry	Charles	Harry

Keep only the columns we need

mother	child	father
Alice	Jim	Bob
Carol	Edward	James
Diana	Charles	Harry

The **JOIN** syntax

To join the tables `Table1` and `Table1`, where the values in the columns `ColumnA` of the first table are the same as the values in `ColumnB` of the second table, we do:

```
SELECT Table1.Column1,... Table2.ColumnN
FROM Table1 JOIN Table2
ON Table1.ColumnA= Table2.ColumnB
```

JOIN mothers and fathers

```
SELECT Table1.Column1,... Table2.ColumnN
FROM Table1 JOIN Table2
ON Table1.ColumnA= Table2.ColumnB
```

Becomes:

```
SELECT Mothers.mother, Fathers.father,
Mothers.child
FROM Mothers JOIN Fathers
ON Mothers.child = Fathers.child;
```

JOIN mothers and fathers

```
SELECT Table1.Column1,... Table2.ColumnN
FROM Table1 JOIN Table2
ON Table1.ColumnA= Table2.ColumnB
```

Becomes:

```
SELECT Mothers.mother, Fathers.father,
Mothers.child
FROM Mothers JOIN Fathers
ON Mothers.child = Fathers.child;
```

Cartesian Product

mother	child	father	child
Alice	Bob	Jim	Bob
Carol	James	Jim	Bob
Diana	Harry	Jim	Bob
Alice	Bob	Edward	James
Carol	James	Edward	James
Diana	Harry	Edward	James
Alice	Bob	Charles	Harry
Carol	James	Charles	Harry
Diana	Harry	Charles	Harry
Alice	Bob	John	Judy
Carol	James	John	Judy
Diana	Harry	John	Judy

```
SELECT *
FROM Mothers JOIN
Fathers
```

Keep only the rows we need

mother	child	father	child
Alice	**Bob**	Jim	**Bob**
Carol	James	Jim	Bob
Diana	Harry	Jim	Bob
Alice	Bob	Edward	James
Carol	**James**	Edward	**James**
Diana	Harry	Edward	James
Alice	Bob	Charles	Harry
Carol	James	Charles	Harry
Diana	**Harry**	Charles	**Harry**
Alice	Bob	John	Judy
Carol	James	John	Judy
Diana	Harry	John	Judy

```
SELECT *
FROM Mothers JOIN
Fathers
ON Mothers.child =
Fathers.child;
```

Keep only the rows we need

mother	child	father	child
Alice	Bob	Jim	Bob
Carol	James	Edward	James
Diana	Harry	Charles	Harry

```
SELECT *
FROM Mothers JOIN
Fathers
ON Mothers.child =
Fathers.child;
```

Keep only the columns we need

mother	father	child
Alice	Jim	Bob
Carol	Edward	James
Diana	Charles	Harry

```
SELECT Mothers.mother,
Fathers.father,
Mothers.child
FROM Mothers JOIN
Fathers
ON Mothers.child =
Fathers.child;
```

Example: pizzas and people

Table **Serves**

pizza	pizzeria
Margherita	Domino's
Margherita	NY Pizza
Mushrooms	Domino's
Mushrooms	NY Pizza
Mushrooms	Pizza Hut
Marinara	Little Caesars
Marinara	Pizza Hut

Table **People**

name	frequents
Alice	Domino's
Alice	NY Pizza
Bob	Domino's
Bob	NY Pizza
Bob	Pizza Hut
Eve	Domino's
Eve	Little Caesars

Example: pizzas and people

Which pizzas can Alice find in the places she frequents?

```
SELECT
FROM Serves JOIN People
ON
```

Example: pizzas and people

Which pizzas can Alice find in the places she frequents?

```
SELECT
FROM Serves JOIN People
ON Serves.pizzeria = People.frequents
```

Example: pizzas and people

Which pizzas can Alice find in the places she frequents?

```
SELECT
FROM Serves JOIN People
ON Serves.pizzeria = People.frequents
WHERE People.name = ‘Alice’ ;
```

Example: pizzas and people

Which pizzas can Alice find in the places she frequents?

```
SELECT DISTINCT Serves.pizza
FROM Serves JOIN People
ON Serves.pizzeria = People.frequents
WHERE People.name = ‘Alice’ ;
```

AS Keyword

With the **AS** keyword, you can give the tables you're using a **temporary** name:

```
SELECT A.a
FROM Foo AS A JOIN Bar AS B
ON A.c = B.d;
```

AS Keyword

With the **AS** keyword, you can give the tables you're using a **temporary** name:

```
SELECT DISTINCT S.pizza
FROM Serves AS S JOIN People AS P
ON S.pizzeria = P.frequents
WHERE P.name = 'Alice' ;
```

Nested **JOIN**

With the **AS** keyword, we can easily nest more **JOIN** instructions:

```
SELECT A.a
FROM Foo AS A JOIN (
   SELECT C.c
   FROM Bar AS C JOIN Qux AS D
   ON ...
   WHERE ...)  AS B
ON A.a = B.c;
```

Note that we can use only variables returned by the inner query!

Exercises

Exercises

1. Find the names and surnames of the editors of each book (return book title, name and surname of the editor).
2. Find the names and surnames of the authors of each book (return book title, name and surname of the author).
3. Find the person and pizzerias in the same city.
4. Find the youngest person in the database using **`JOIN`** instead of **`IN`** . Hint: you can rename columns like you rename tables, e.g. **`SELECT AVG(x) AS 'avg'`**
5. Find all pizzerias frequented by at least one person under the age of 18.

Exercises

6. Sort the countries by the population of their cities (biggest first)
7. Return a table containing each country and its official language
8. Return a table containing each country associated with its capital city and the district of the capital city
   e.g. UK, London, England
9. Return a table containing the capital city district and the official language of each country

Remember to select the **world** database