

# Data Management Systems

## PostgreSQL

Matteo Devigili

June 9<sup>th</sup>, 2023

# SET UP

## What we have

location		person		car	
id	bigserial	id	BIGSERIAL	id	bigserial
country	varchar(50)	first_name	VARCHAR(50)	car_make	varchar(50)
city	varchar(50)	last_name	VARCHAR(50)	car_model	varchar(50)
street_name	varchar(50)	email	VARCHAR(50)	car_year	int
street_number	int	gender	VARCHAR(50)	price	numeric
postal_code	varchar(50)	dob	DATE		

# SET UP

## What we need

location		person		car	
<b>id</b>	<b>bigserial</b>	<b>id</b>	<b>BIGSERIAL</b>	<b>id</b>	<b>bigserial</b>
country	varchar(50)	first_name	VARCHAR(50)	car_make	varchar(50)
city	varchar(50)	last_name	VARCHAR(50)	car_model	varchar(50)
street_name	varchar(50)	email	VARCHAR(50)	car_year	int
street_number	int	gender	VARCHAR(50)	price	numeric
postal_code	varchar(50)	dob	DATE		
person_id	bigint	car_id	bigint		

# SET UP

## Relations

location		person		car	
<b>id</b>	<b>bigserial</b>	<b>id</b>	<b>BIGSERIAL</b>	<b>id</b>	<b>bigserial</b>
country	varchar(50)	first_name	VARCHAR(50)	car_make	varchar(50)
city	varchar(50)	last_name	VARCHAR(50)	car_model	varchar(50)
street_name	varchar(50)	email	VARCHAR(50)	car_year	int
street_number	int	gender	VARCHAR(50)	price	numeric
postal_code	varchar(50)	dob	DATE		
person_id	bigint	car_id	bigint		

# FOREIGN KEY

- ▶ *FOREIGN KEY*: specifies that the values in a column (or a group of columns) must match the values appearing in some row of another table

Postgre - Constraints

# FOREIGN KEY

## many-to-many example

```
CREATE TABLE courses (  
    SMM_Code text PRIMARY KEY  
    Lecturer text  
    Term text);
```

```
CREATE TABLE student (  
    Student_id int PRIMARY KEY  
    firs_name text  
    last_name text);
```

```
CREATE TABLE grades (  
    SMM_Code text REFERENCES courses,  
    Student_id text REFERENCES student,  
    Grade varchar(1),  
    PRIMARY KEY (SMM_Code, Student_id )  
);
```

# FOREIGN KEY

## many-to-many example

```
CREATE TABLE courses (  
    SMM_Code text PRIMARY KEY  
    Lecturer text  
    Term text);
```

```
CREATE TABLE student (  
    Student_id int PRIMARY KEY  
    firs_name text  
    last_name text);
```

```
CREATE TABLE grades (  
    SMM_Code text REFERENCES courses,  
    Student_id text REFERENCES student,  
    Grade varchar(1),  
    PRIMARY KEY (SMM_Code, Student_id )  
);
```

# FOREIGN KEY

many-to-many example

```
CREATE TABLE courses (
```

```
  SMM_Code text PRIMARY KEY
```

```
  Lecturer text
```

```
  Term text);
```

```
CREATE TABLE student (
```

```
  Student_id int PRIMARY KEY
```

```
  firs_name text
```

```
  last_name text);
```

```
CREATE TABLE grades (
```

```
  SMM_Code text REFERENCES courses,
```

```
  Student_id text REFERENCES student,
```

```
  Grade varchar(1),
```

```
  PRIMARY KEY (SMM_Code, Student_id )  
);
```



# SQL

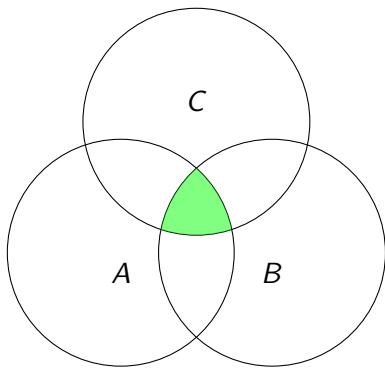
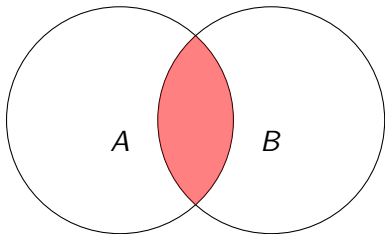
## Recap of commands

- ▶ *ALTER TABLE*: to change the definition of an existing table
- ▶ *DELETE FROM WHERE*: to delete rows that satisfy a condition from a table
- ▶ *UPDATE SET WHERE*: to change the values of the specified columns in all rows that satisfy a condition

PostgreSQL

# INNER JOIN

Venn diagram



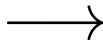
# INNER JOIN

A

a
b
c
d

B

c
d
e
f



c
d

# INNER JOIN

Joining tables on column **id** of table **A**, and column **A\_id** of table **B**:

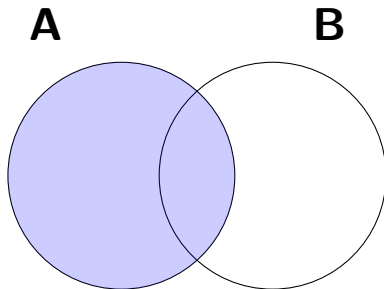
A	
id	year
1	2008
2	2010
3	2012

B	
A_id	city
1	Beijing
3	London
5	Rio de Janeiro

C			
id	year	A_id	city
1	2008	1	Beijing
3	2012	3	London

# LEFT OUTER JOIN

Venn diagram



# LEFT OUTER JOIN

LEFT (OUTER) JOIN on column id of table **A**, and column A\_id of table **B**:

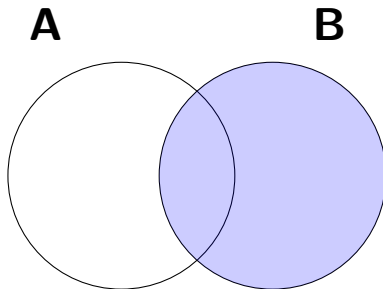
A	
id	year
1	2008
2	2010
3	2012

B	
A_id	city
1	Beijing
3	London
5	Rio de Janeiro

C			
id	year	A_id	city
1	2008	1	Beijing
2	2010		
3	2012	3	London

# RIGHT OUTER JOIN

Venn diagram



# RIGHT OUTER JOIN

RIGHT (OUTER) JOIN on column id of table **A**, and column A\_id of table **B**:

A	
id	year
1	2008
2	2010
3	2012

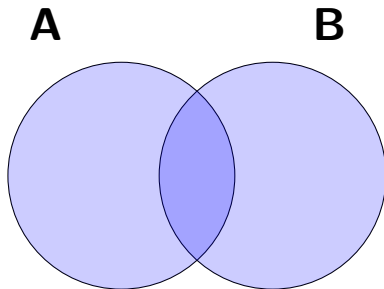
B	
A_id	city
1	Beijing
3	London
5	Rio de Janeiro

C			
id	year	A_id	city
1	2008	1	Beijing
3	2012	3	London
		5	Rio de Janeiro



# FULL OUTER JOIN

Venn diagram



# FULL OUTER JOIN

FULL (OUTER )JOIN on column id of table **A**, and column A\_id of table **B**:

A	
id	year
1	2008
2	2010
3	2012

B	
A_id	city
1	Beijing
3	London
5	Rio de Janeiro

C			
id	year	A_id	city
1	2008	1	Beijing
2	2010		
3	2012	3	London
		5	Rio de Janeiro

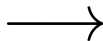
# CROSS JOIN

A

1
2

B

a
b



1	a
1	b
2	a
2	b

# SQL - Joins

## Recap

- ▶ *(INNER) JOIN ... ON*: For each row R1 of A, the joined table has a row for each row in B that satisfies the join condition with R1.
- ▶ *LEFT (OUTER) JOIN ... ON*: First, an inner join is performed. Then, for each row in A that does not satisfy the join condition with any row in B, a joined row is added with null values in columns of B. Thus, the joined table always has at least one row for each row in A.
- ▶ *RIGHT (OUTER) JOIN ... ON*: First, an inner join is performed. Then, for each row in B that does not satisfy the join condition with any row in A, a joined row is added with null values in columns of A. This is the converse of a left join: the result table will always have a row for each row in B.

## PostgreSQL - Joined Tables

# SQL - Joins

## Recap

- ▶ *FULL (OUTER) JOIN ... ON*: First, an inner join is performed. Then, for each row in A that does not satisfy the join condition with any row in B, a joined row is added with null values in columns of B. Also, for each row of B that does not satisfy the join condition with any row in A, a joined row with null values in the columns of A is added.
- ▶ *CROSS JOIN*: For every possible combination of rows from A and B (i.e., a Cartesian product), the joined table will contain a row consisting of all columns in A followed by all columns in B.

## PostgreSQL - Joined Tables

# References

- ▶ Obe, Regina O., and Leo S. Hsu. PostgreSQL: Up and Running: a Practical Guide to the Advanced Open Source Database. "O'Reilly Media, Inc.", 2017.
- ▶ PostgreSQL 13 Documentation  
<https://www.postgresql.org/docs/12/index.html>
- ▶ SQL tutorial <https://www.sqltutorial.org>