Corso SQL e PostgreSQL

Lezione 1 - Introduzione

Link video lezione: https://www.youtube.com/watch?v=5hzZtqCNQKk

Cosa è un database: Una collezione strutturata di informazioni

La struttura preferita è in **tabella**, costituita a sua volta da **colonne** (che determina l'attributo) e **righe** (che contengono il dato).

Ci sono tanti tipi di databases: sequenziali, ad oggetti, non sequenziali, ecc…

Quelli sequenziali sono quelli più famosi.

Es.

Tabella guidatore --- Tabella automobili // Sono **legabili** da una relazione

---

Lezione 2 - Cosa è SQL

SQL: Structured Query Language

Un linguaggio per effettuare delle domande ad un database.

E' il linguaggio più usato per effettuare query ad un database relazionale

---

Esempio di query in SQL

SELECT colonna FROM tabella // seleziona una colonna da una tabella

Tabella Person

id	first_name	last_name	gender	age
1	anne	smith	female	22
2	…	…	…	…
3	…	…	…	…

Colonne e righe (**columns** and **rows**)

Lezione 3 - DBMS: DataBase Management System

Il database è la struttura virtuale di dati, SQL è il linguaggio per richiamare questi dati, ma serve un **software** per gestire i databases…il DBMS

Oltre a **Postgres** ne esistono molti altri di questi DBMS:

Es.: **MySQL, Microsoft SQL, SQLlite,** ecc…

N.B. Non preoccuparti per i vari dbms, basta imparare sql, le differenze sono poche. (almeno, spero)

Lezione 4 - Installazione DBMS (Postgres)

(Segui installazione video lezione e tutorial Lorenzo)

Lezione 5 - Basi Psql Shell

Psql: la shell che permette di gestire i databases.

È possibile anche usare l'interfaccia grafica.

N.B. Meglio usare l'**UPPERCASE**

Utente username: postgres

Password: quella indicata in fase di installazione

Comandi:

\? : lista dei comandi

\l : lista databases

\q : esce fuori dalla psql shell

\! cls : pulire la shell

Creare un database: CREATE DATABASE nome_database ;

Lezione 6 - Connettersi ad un database creato

\c[onnect] {[DBNAME|- USER|- HOST|- PORT|-] | conninfo}

\c nome_database (es. \c test)

Lezione 7 - Creare Tabelle (senza Costrains**)**

Sintassi:

CREATE TABLE name_table (

Column name + data type + costrains (if any)

)

Esempio:

CREATE TABLE person (

Id int,

first_name VARCHAR(50),

last_name VARCHAR(50),

gender VARCHAR(6),

date_of_birth DATE

)

1 tabella, 5 colonne, 1 riga

N.B. in VACCHAR è sottointeso il tipo di dato, ovvero text (o stringa)

N.B. ricorda il **SEMI-COLON**!!! **;**

Se la creazione è andata a buon fine comparirà **CREATE TABLE**

\d : per avere una descrizione delle tabelle e degli **schemas**

\dt : per vedere **solo** le tabelle

\d nome_tabella : descrizione tabella

Lezione 8 - Creare Tabelle (con Costrains)

Esempio:

```
CREATE TABLE person (
        Id BIGSERIAL NOT NULL PRIMARY KEY,
        first_name VARCHAR(50) NOT NULL,
        last_name VARCHAR(50) NOT NULL,
        gender VARCHAR(6) NOT NULL,
        date_of_birth DATE NOT NULL,
        email VARCHAR(150)
);
```

PRIMARY KEY: è il primo parametro da inserire

N.B. Con **BIGSERIAL** avremo, ad ogni aggiunta di una persona, un id che si auto-**incrementa** di 1

NOT NULL: specifica che quel campo non può rimanere vuoto.

(Es. vuoi aggiungere una persona? Non possono mancare i campi con **NOT NULL**).

DROP TABLE nome_table; : cancellare **tutto ciò che c'è dentro** una tabella (attento a questa operazione)

La tabella person neogenerata sarà di tipo **sequence**.

Lezione 9 - Inserire dati in una tabella

Esempio:

```
INSERT INTO nome_tabella (
        First_name,
        last_name,
        gender,
        date_of_birth
)
VALUES ('Anne','Smith','FEMALE', '1988-01-09');
```

INSERT 0 1 : se l'inserimento sarà andato a buon fine comparirà questa scritta.

Lezione 9 - SELECT FROM

SELECT nome_colonna FROM nome_tabella;

Esempi:

SELECT * FROM person;

SELECT first_name FROM person;

SELECT first_name, last_name FROM person;

ecc…

Lezione 10 - Aggiungere Dati in modo massivo

Usare **Mockaroo** è utile

Usare VSCode

\i FILE : Comando per eseguire un file esterno in sql

Non è possibile riempire la tabella perché è stato aggiunto un campo che prima non c'era.

Droppiamola.

Rifacciamola aggiungendo in automatico un id incremento automatico (BIGSERIAL).

Lezione 11 - Sorting

Mettere in ordine in base al contenuto delle colonne

Sintassi:

SELECT quale_colonna FROM nome_tabella ORDER BY attributo_colonna ASC/DESC;

Esempio 1: Mettere in ordine per una colonna

SELECT * FROM person ORDER BY country_of_birth ASC;

Esempio 2: Mettere in ordine per 2 colonne

SELECT * FROM person ORDER BY id, country_of_birth ASC;

Lezione 12 - Rimuovere duplicati

Sintassi:

SELECT DISTINCT quale_colonna FROM nome_tabella ORDER BY attributo_colonna ASC/DESC;

Es.: SELECT DISTINCT country_of_birth FROM person ORDER BY country_of_birth ASC;

Selezione dalla tabella person le righe con una country per tipo ordinate in modo ascendente (Da A a Z).

Lezione 13 - WHERE CLAUSE

WHERE filtra le colonne in base ad una condizione (clausola).

Es.: con **AND**

SELECT * FROM person WHERE gender = 'FEMALE' AND country_of_birth = 'Poland';

Es.: con **AND** e **OR**

SELECT * FROM person WHERE gender = 'FEMALE' AND (country_of_birth = 'Poland' OR country_of_birth = 'China');

Es. Con **doppio AND** e **OR**

SELECT * FROM person WHERE gender = 'FEMALE' AND (country_of_birth = 'Poland' OR country_of_birth = 'China') AND last_name = 'Pietersma';

Accetta anche l'**AND** per definire più **condizioni.**

Lezione 14 - Operatori aritmetici e di comparazione

Comparare Numeri

SELECT 1 = 1;

// true

SELECT 1 < 1;

// false

SELECT 1 <= 2;

// true

SELECT 1<>2; (**<>** Non uguale)

// true

Ecc…

Comparare Stringhe

SELECT ‘stringa’ = ‘stringa’;

// true

SELECT ‘stringa’ = ‘STRINGA;

// false

SELECT ‘stringa’ <> ‘STRINGA;

// true

Lezione 15 - LIMIT, OFFEST and FETCH

Sintassi:

SELECT * FROM person LIMIT numero;

Esempio:

SELECT * FROM person LIMIT 10;

// Stampa solo 10 righe

Sintassi:

SELECT * FROM person OFFSET numero LIMIT numero;

Esempio:

SELECT * FROM person OFFSET 5 LIMIT 10;

// Stampa 10 righe dalla riga 5

Sintassi:

SELECT * FROM person OFFSET numero FETCH numero ROWS ONLY;

Esempio:

SELECT * FROM person OFFSET 5 FETCH 5 ROWS ONLY;

// Stampa 10 righe dalla riga 5

Lezione 16 - IN keyword

SELECT * FROM person WHERE country_of_birth = 'Brasil' AND country_of_birth = 'France' AND country_of_birth = 'China';

V

SELECT * FROM person WHERE country_of_birth **IN** ('China','Brasil','France');

IN permette di uguagliare in base ai valori della colonna nella parentesi

---

Lezione 17 - BETWEEN

Esempio:

SELECT * FROM person WHERE date_of_birth BETWEEN DATE '2021-01-01' AND '2021-21-31';

N.B. DATE indica il tipo di dato, che in questo caso è date

Lezione 18 - LIKE

LIKE è come un cercatore di sotto stringa

Esempio:

SELECT * FROM person WHERE email LIKE '%.com';

// Ovvero cerca email che finiscono per .com

N.B. **%** indica **da**

Esempio 2:

SELECT * FROM person WHERE email LIKE '%google.%';

// cerca righe che hanno email che hanno “google.” dentro.

Esempio 3:

SELECT * FROM person WHERE email LIKE '_____@%';

// cerca email in person dove si sono tot (nel caso sopra sono 5 _, ovvero 5 caratteri) caratteri dentro prima della @

N.B. Oltre a LIKE esiste anche **ILIKE**, il contrario.

Lezione 19 - GROUP BY

Raggruppa per un **attributo** le mie **rows** (righe)

SELECT country_of_birth**, COUNT (*)** FROM person **GROUP BY** country_of_birth;

// Seleziona una Colonna delle tabella e raccoglie tutti i dati per quell’attributo, contandone il numero

SELECT country_of_birth, COUNT(*) FROM person GROUP BY country_of_birth ORDER BY country_of_birth;

// Lo stesso, ma mi riordina in ordine alfabetico.

N.B. **COUNT(*)** è una funzione che fa un conto per ogni riga(guarda nella documentazione)

Lezione 19 - HAVING

SELECT country_of_birth, COUNT(*) FROM person GROUP BY country_of_birth HAVING COUNT(*) > 5 ORDER BY country_of_birth;

// Con HAVING specifichiamo delle righe che hanno una condizione (in questo caso un COUNT con un valore superiore a 5).

Lezione 20 - CAR TABLE

MAX(number):

SELECT MAX(price) FROM car;

// Prezzo Massimo

MIN(number):

SELECT MIN(price) FROM car;

//Prezzo minimo

AVG(number):

SELECT AVG(price) FROM car;

//Prezzo medio (media aritmetica)

ROUND(number):

SELECT ROUND(AVG(price)) FROM car;

//Prezzo medio (media aritmetica) arrotondato

SELECT make, model, MIN(price) FROM car GROUP BY make, model;

// Raggruppa le auto per nome e modello, stampando quello, di quella marca e modello con il prezzo minimo

SUM:

SELECT make, SUM(price) FROM car GROUP BY make;

// Seleziona tutte le auto per marca ed effettua la somma dei prezzi

SELECT make, MAX(price) FROM car GROUP BY make ORDER BY MAX(price);

// Riordino per prezzo massimo le auto di quella marca

Lezione 21 - Operatori Aritmetici

Alcuni esempi:

SELECT 1 + 1;

// 2

SELECT 10 + 1;

// 9

SELECT 10^2;

// 100

SELECT 5!;

// 125

SELECT 1 * 2 + 3;

// 5

SELECT 10%3;

// 1

Lezione 22 - Operatori Aritmetici pt.2

SELECT id, make, model, price, **price * 0.10** FROM car;

// Creare una nuova colonna che aggiunge il prezzo con il 10% di sconto

SELECT id, make, model, price, **ROUND(price * 0.10, 2)** FROM car;

// Creare una nuova colonna che aggiunge il prezzo con il 10% di sconto arrotondato al percento.

SELECT id, make, model, price, ROUND(price * 0.10, 2), **ROUND(price - price * 0.10, 2)** FROM car;

// Creare una nuova colonna che aggiunge il prezzo con il 10% di sconto arrotondato al percento e aggiunge un'altra colonna con lo sconto applicato

N.B. La creazione delle neo-colonne create (10%) è volatile, non permane nella tabella.

Lezione 23 - Alias

Rinominare le neo-colonne create nel capitolo precedente usando **AS nome_neo_colonna**

Esempio:

SELECT id, make, model,

Price **AS original_price**,

ROUND(price * 0.10, 2) **AS ten_percent**,

ROUND(price - price * 0.10, 2) **AS discount_after_ten_percent**

FROM car

;

Lezione 24 - COALESCE

COALESCE: funzione che restituirà il primo valore valido interno

Esempi:

SELECT COALESCE(null, null, 1);

// 1

SELECT COALESCE(null, null, 1, 10);

// 1

SELECT **COALESCE(email, 'Email not provided')** FROM person;

// Restituirà I valori della colonna email, ma dove un utente non avrà una email verrà restituito 'Email not provided'

Lezione 25 - NULLIF

SELECT NULLIF(A, B)

// Se A = B non verrà restituito nulla

// Se A != B verrà restituito A

Un sistema che, associato al COALESCE, permette di amministrare le **divisioni per 0**, senza suscitare errori

Esempio:

SELECT COALESCE(

10 / NULLIF(0,0)

, 0)

// 0

Lezione 26 - Timestamps e Dates

Esempi:

SELECT NOW();

// 2022-01-17 15:43:05.311169+01

SELECT NOW()::DATE;

// 2022-01-17

SELECT NOW()::TIME;

// 15:44:16.874759

Lezione 27 - Aggiungere e sottrarre date

Sottrazione/Addizione date:

SELECT NOW() – INTERVAL ‘1 YEAR’;

// Data oggi – un anno

SELECT NOW() + INTERVAL ‘3 MONTHS’;

// Data oggi + 3 mesi

SELECT **(NOW() + INTERVAL ‘3 MONTHS’)::DATE** ;

// Data oggi + 3 mesi (solo **dd-mm-yy**)

Lezione 28 - Estrarre campi dalle date

SELECT **EXTRACT(** YEAR FROM NOW() **)** ;

// Estrae unicamente l’anno dalla data attuale

SELECT **EXTRACT(** CENTURY FROM NOW() **)** ;

// Estrae unicamente il secolo della data attuale

Lezione 29 - AGE FUNCTION

Sintassi:

AGE(NOW(), date_of_birth): permette di definire l’età in base ai dati inseriti

Esempio:

SELECT first_name, last_name, gender, country_of_birth, date_of_birth, **AGE(NOW(), date_of_birth)** AS age FROM person;

Lezione 30 - PRIMARY KEYS (pt.1)

Il **Primary Key** (**PK**) è il valore che serve ad' identificare univocamente la row di riferimento

---

Lezione 31 - PRIMARY KEYS (pt.2)

"nextval('person_id_seq'::regclass)"

// Autoincremento per unità

N.B. A causa del **primary key univoco** non è possibile **inserire** ulteriori entità con **id uguale.**

Rimozione di un costraint:

Sintassi:

ALTER TABLE nome_table DROP **COTRAINT** nome_costraint;

Esempio:

ALTER TABLE person DROP COTRAINT person_pkey;

// Questo permetterà l'inserimento di entità con id uguale…ma a che prò?

Aggiunta di un costraint:

Sintassi:

ALTER TABLE nome_table ADD **PRIMARY KEY(colonna1, colonna2, ecc…)**;

Esempio:

ALTER TABLE person ADD **PRIMARY KEY(id)**;

// Questo permetterà di reintegrare il primary key sull'id

// ALTER TABLE

--- N.B. Prima di restaurare il costraint primary key bisogna eliminare i duplicati eventuali creati in post

DELETE FROM person WHERE id=1;

// Cancellazione di righe con id=1

Lezione 32 – COSTRAINTS unici

(Come creare **costraints personalizzati**)

N.B. **UNIQUE COSTRAINTS** =/= **PRIMARY KEY**

Problema:

E se ci fossero utenti con email uguali ma id distinti e volessimo mandare l'email ad una sola di esse?

Sintassi:

ALTER TABLE nome_table ADD COSTRAINT nome_nuovo_costraint_da_aggiungere **UNIQUE(colonna)**

UNIQUE(colonna1, colonna2, ecc…)

Esempio:

ALTER TABLE person ADD COSTRAINT unique_email_address **UNIQUE(email);**

// In questo caso, visto che nella tabella ci sono 2 persone con stessa email, non possiamo porre questo costraint aggiuntivo, per poterlo fare dobbiamo o cancellare una delle persone oppure cambiare l'email di una delle 2

Lezione 33 – Check COSTRAINTS

Aggiungere un **costraints** in base ad un valore **booleano**

ALTER TABLE person ADD COSTRAINT gender_costraint CHECK(gender = 'Female' OR gender = 'Male');

Rivedere Lezioni 30, 31, 32, 33 sui COSTRAINTS

Lezione 34 - Deleting Records

Per la cancellazione di un item della lista è sempre conveniente usare come riferimento una colonna con il PK (ovvero l'id), così da non incorrere in ambiguità di cancellazione

Sintassi:

DELETE FROM nome_tabella;

// Si cancella tutto dalla tabella, rimarrà vuota

// N.B. al reinserimento di valori, si partirà dall'id seguente a ll'ultimo presente nella lista cancellata

Esempio:

DELETE FROM person WHERE id = 101;

// Cancella l'item con id univoco pari a 101

DELETE FROM person WHERE id = 101 AND id=102;

// Cancella gli item con id univoco pari a 101 e a 102

Lezione 35 - Updating Records

Sintassi:

UPDATE nome_tabella SET nome_colonna='valore_aggiornato';

// Questo aggiornerà con il nuovo valore, nella tabella di riferimento, tutti i valori della colonna definita

Esempio 1:

UPDATE person SET email='email_aggiunta@gmail.com' WHERE id=1;

// Aggiornerà da vuota il campo email con l'id definito, **solo quello**

Esempio 2:

UPDATE person SET first_name = 'nome_modificato', email='email_aggiunta@gmail.com' WHERE id=1;

// Aggiornerà da vuota il campo email con l'id definito, **solo quello**

Lezione 36 - ON CONFLICT DO NOTHING

Manipolazione dell'errore nel qual caso in cui se ne presentino

Problema:

Voglio inserire un nuovo item in una tabella

Questo nuovo item ha un id uguale ad un altro presente in lista

// Dal tentativo di inserimento **scaturirà un errore**

Sintassi:

ON CONFLICT (colonna_presso_cui_si_genera_il_conflitto) DO NOTHING

// Se tenterò di inserire un item con id uguale con un altro item già presente, e che quindi **va in conflitto**,

non comparirà un messaggio di errore, invece **non accadrà nulla** (**INSERT 0 0**)

Esempio:

ON CONFLICT (id) DO NOTHING

N.B. l'ON CONFLICT non funziona su colonne senza COSTRAINTS

Lezione 37 - ON CONFLICT DO UPDATING

Problema:

E se un utente registrato volesse modificare la propria email?

In tal caso è possibile manipolare la gestione dell'errore usando DO UPDATING settando l'unico parametro che ci interessa gestire in caso di conflitto, escludendolo da esso

Sintassi:

…insert new item

ON CONFLICT (colonna_PK) DO UPDATE SET nome_colonna = 'EXCLUDED. nome_colonna';

Esempio:

…insert new person

ON CONFLICT (id) DO UPDATE SET email = 'EXCLUDED.email ;

// Ci sarà un conflitto e una non modifica per tutte le colonne legate alla row con quell'id (definito dall'insert dell'item) ma non per l'email, che verrà modificata.

Lezione 37 - FOREIGN KEYS e RELATIONSHIP

FOREIGN KEYS: Permettono di relazionare 2 tabelle (**relationship**)

Tabella Persone ----- Tabella Auto

- Ci sono persone con un'auto
- Ci sono persone senza auto
- Le auto possono appartenere ad una sola persona

Le FOREIGN KEYS sono delle colonne che, inserite in una tabella, permettono di definire una relazione con un'altra tabella

Esempio:

Tabella Persona

Id = 1,

Name = 'John',

Surname = 'Smith',

has_auto true (ha un **id che lo collega*** ad un auto dell'altra tabella)

Id = 2,

Name = 'Tomas',

Surname = 'Enderson',

has_auto false (non ha **un id che lo collega** ad un auto dell'altra tabella)

*Questo identificativo è il **FOREIGN KEY**

// La prima persona (id=1) della tabella Persona, ha un id che lo collega univocamente ad un auto, la seconda no.

Lezione 38 – Aggiungere un FOREIGN KEY

Tabella person

```
car_id BIGINT REFERENCES car (id),
/* Non mettiamo not null perchè alcune persone possono non avere un auto, quindi
avranno il car_id vuoto - REFERENCES ci permette di riferirci alla colonna id della
tabella car */
UNIQUE (car_id) /* Aggiungiamo un COSTRAINT AL NOSTRO car_id */
```

Indici:

"person_pkey" PRIMARY KEY, btree (id)

"person_car_id_key" UNIQUE CONSTRAINT, btree (car_id)

Vincoli di integritÓ referenziale

"person_car_id_fkey" FOREIGN KEY (car_id) REFERENCES car(id)

Tabella car

Id a cui si riferisce il **car_id** di **person**

Lezione 39 - Modificare un FOREIGN KEY

Sintassi:

UPDATE person SET car_id = 1 WHERE id = 1;

// Aggiorna la tabella person, qui , nella persona con id uguale a 1, devi settare il car_id ad 1,

SELECT * FROM person;

id | first_name | last_name | gender | email | date_of_birth | country_of_birth | car_id

----+------------+-----------+--------+---------------------+---------------+------------------+--------

1 | Fernanda | Beardon | Female | fernandab@is.gd | 1953-10-28 | Comoros |

2 | Omar | Colmore | Male | | 1921-04-03 | Finland |

3 | John | Matuschek | Male | john@feedburner.com | 1965-02-28 | England |

SELECT * FROM CAR;

```
 id |    make    |  model   |  price
----+------------+----------+----------
  1 | Land Rover | Sterling | 87665.38
  2 | GMC        | Acadia   | 17662.69
```

UPDATE person SET car_id = 1 WHERE id = 1;

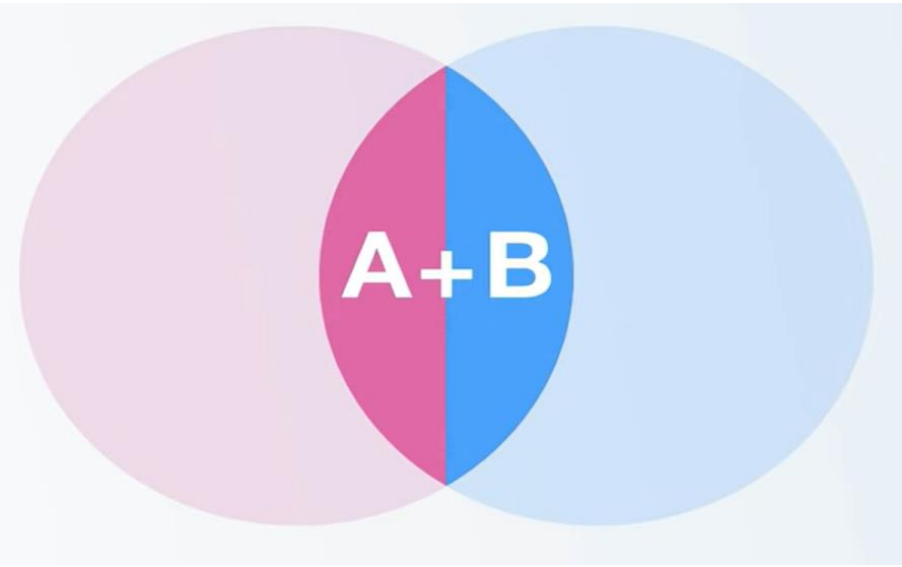
SELECT * FROM person;

```
 id | first_name | last_name | gender |       email       | date_of_birth | country_of_birth | car_id
----+------------+-----------+--------+-------------------+---------------+------------------+--------
  2 | Omar       | Colmore   | Male   |                   | 1921-04-03    | Finland          |
  3 | John       | Matuschek | Male   | john@feedburner.com | 1965-02-28    | England          |
  1 | Fernanda   | Beardon   | Female | fernandab@is.gd   | 1953-10-28    | Comoros          |      1
```

1 | Fernanda | Beardon ----- 1 | Land Rover | Sterling

Hanno una relazione

Lezione 40 - INNER JOINS



Inner Join permette di, in caso di FOREIGN KEYS, di avere una terza tabella risultante che esplica la relazione

Sintassi:

SELECT * FROM tabella_A JOIN tabella_B ON tabella.sua_colonna_da_unire = tabellaB.sua_colonna;

Esempio:

SELECT * FROM person JOIN car ON person.car_id = car.id;

// Appaierà le tabelle in base al FOREIGN KEYS

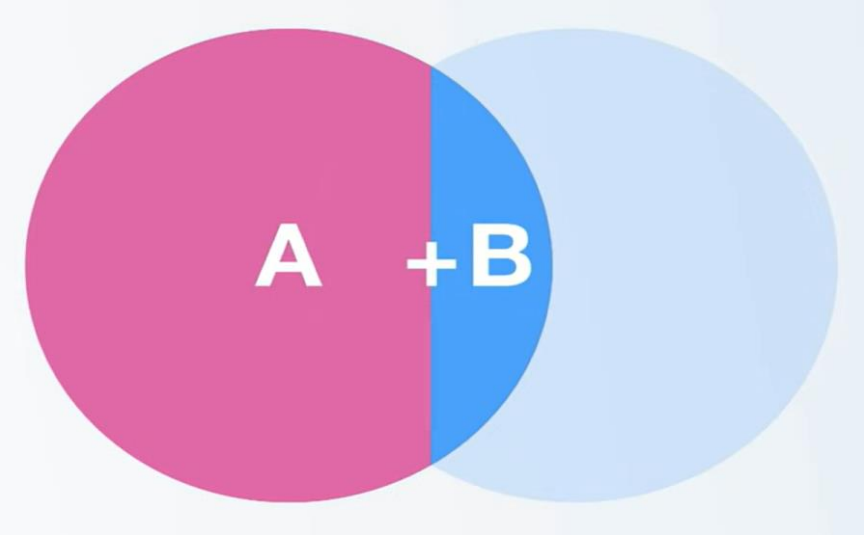
\x : Visualizzazione **espansa** delle tabelle

Risultato INNER JOIN:

SELECT * FROM person JOIN car ON person.car_id = car.id;

```
 id | first_name | last_name | gender |       email       | date_of_birth | country_of_birth | car_id | id |    make    |  model   |  price
----+------------+-----------+--------+-------------------+---------------+------------------+--------+----+------------+----------+----------
  1 | Fernanda   | Beardon   | Female | fernandab@is.gd   | 1953-10-28    | Comoros          |      1 |  1 | Land Rover | Sterling | 87665.38
  3 | John       | Matuschek | Male   | john@feedburner.com | 1965-02-28  | England          |      2 |  2 | GMC        | Acadia   | 17662.69
(2 righe)
```

Lezione 41 - LEFT JOINT



Left Join permette di, in caso di FOREIGN KEYS, di avere una terza tabella risultante che esplica la relazione, **includendo** anche gli elementi che hanno **una non relazione**

Sintassi:

SELECT * FROM tabella_A LEFT JOIN tabella_B ON tabella.sua_colonna_da_unire = tabellaB.sua_colonna;

Esempio:

SELECT * FROM person LEFT JOIN car ON person.car_id = car.id;

// Appaierà le tabelle in base al FOREIGN KEYS, mettendo anche le righe non appaiate

Risultato LEFT JOIN:

SELECT * FROM person LEFT JOIN car ON person.car_id = car.id;

```
 id | first_name | last_name | gender |       email       | date_of_birth | country_of_birth | car_id | id |    make    |  model   | price
----+------------+-----------+--------+---------------------+---------------+------------------+--------+----+------------+----------+----------
  1 | Fernanda   | Beardon   | Female | fernandab@is.gd     | 1953-10-28    | Comoros          |      1 |  1 | Land Rover | Sterling | 87665.38
  3 | John       | Matuschek | Male   | john@feedburner.com | 1965-02-28    | England          |      2 |  2 | GMC        | Acadia   | 17662.69
  2 | Omar       | Colmore   | Male   |                     | 1921-04-03    | Finland          |        |    |            |          |
(3 righe)
```

N.B.: SELECT * FROM person LEFT JOIN car ON person.car_id = car.id **WHERE car.* IS NULL**;

// 2 | Omar | Colmore | Male | | 1921-04-03 | Finland | | | | |

IS NULL : Serve per visualizzare la riga con il relativo campo vuoto.

Lezione 42 - Deleting FOREIGN KEYS

Cancellare FOREIGN KEYS

Inserire persona test:

```
insert into person (id, first_name, last_name, gender, email, date_of_birth,
country_of_birth) values (9000,'Bob', 'Suio', 'Male', null, '1921-04-03',
'Finland');
insert into car (id, make, model, price) values (9000, 'Panda', 'Sterling',
'87665.38');
```

Inserire auto test:

```
insert into car (id, make, model, price) values (9000, 'Panda', 'Sterling',
'87665.38');
```

Aggiornare una foreign Keys così da appaiarli

UPDATE person SET car_id = 9000 WHERE id = 9000;

id | make | model | price

------+-------+----------+----------

9000 | Panda | Sterling | 87665.38

id | first_name | last_name | gender | email | date_of_birth | country_of_birth | car_id

------+------------+-----------+--------+-------+---------------+------------------+--------

9000 | Bob | Suio | Male | | 1921-04-03 | Finland | 9000

Adesso Bob ha un'auto con l'id pari a 9000.

Proviamo a cancellare l'auto:

DELETE FROM car WHERE id = 9000;

ERRORE: l'istruzione UPDATE o DELETE sulla tabella "car" viola il vincolo di chiave esterna "person_car_id_fkey" sulla tabella "person"

DETAIL: La chiave (id)=(9000) è ancora referenziata dalla tabella "person".

// La cancellazione è inibita perché c'è una foreign key che viene usata dalla persona, e la reference rimane nella person

2 soluzioni:

1) Cancellare utente per cancellare l'auto
2) Cambiare il car_id (in **NULL**) dell'utente

Lezione 43 - SEQUENCES

Tabella "public.person"

Colonna	Tipo	Ordinamento	Può essere null	Default
id	bigint		not null	**nextval('person_id_seq'::regclass)***
first_name	character varying(50)		not null	

(…)

Indici:

"person_pkey" PRIMARY KEY, btree (id)

(…)

* SELECT * FROM **person_id_seq;**

last_value | log_cnt | is_called

----------+----------+----------

3 | 30 | t

(1 riga)

SELECT nextval('person_id_seq'::regclass);

// Visualizzerà il valore dopo, **inserendolo**

Quindi, se si inserisse un nuovo item, questo avrà un id **+ 1**

E per resettarlo (o meglio, alterarlo?)

Sintassi:

ALTER SEQUENCE nome_tabella_id_seq RESTART WITH valore_desiderato;

Esempio:

ALTER SEQUENCE person_id_seq RESTART WITH 4;

Lezione 44 - EXTENSIONS

(esempio con uuid)

Controllare lista **EXTENSIONS** installate

SELECT * FROM pg_available_extensions;

Installare una **EXTENSION**

Sintassi:

CREATE EXTENSION IF NOT EXISTS “nome-estensione”;

Esempio:

CREATE EXTENSION IF NOT EXISTS “uuid-ossp”;

\df : permette di vedere quali funzioni sono aggiunte con le estensioni

---

Usare una funzione introdotta da una **EXTENSION**

SELECT nome_funzione;

SELECT uuid_generate_v4();

// Funzione utile a generare id casuali
