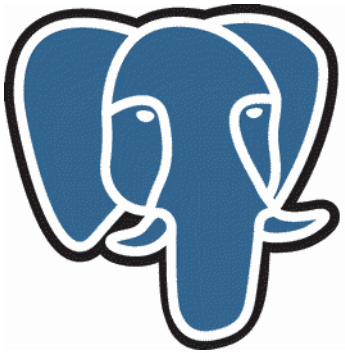
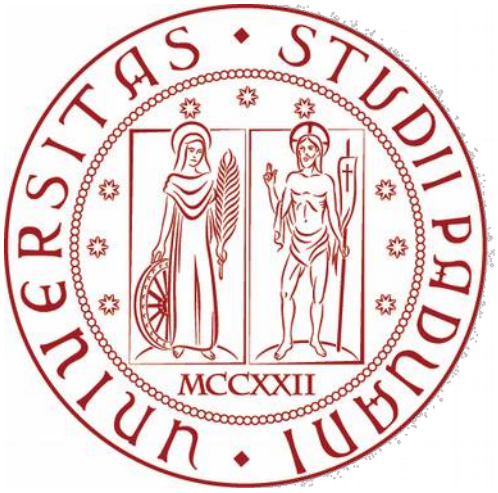


PostgreSQL

the world's most advanced open source database

#PostgreSQL-IT

9 Giugno 2015 – Corso di Basi di Dati



UNIVERSITÀ DEGLI STUDI DI PADOVA

Presentazioni

Denis Gasparin

Senior DBA and Web Developer

- Sviluppo di soluzioni software basate su PostgreSQL
- Analista e Database Administrator
- Contributor del driver PDO PostgreSQL per PHP
- Socio e segretario di IT-PUG

PostgreSQL: chi è costui?

- **1 Maggio 1995**
  - Postgres95 V0.01
- **6 Major Release dal 1995**
  - PostgreSQL95
  - PostgreSQL 1.0
  - PostgreSQL 6, 7, 8, 9
  - 23 Minor Release
- **Una versione all'anno**

Pg(SQL) = S^3

- Nota funzione matematica coniata da...
  - Michael Stonebraker?
  - Andrew Yu e Jolly Chen?
  - … Denis Gasparin :-D

- **PostgreSQL** è la moltiplicazione di tre fattori:
  - **Semplicità**
  - **Scalabilità**
  - **Sicurezza**

Semplicità

- Pacchetti di installazione disponibili per
  - Tutte le maggiori distribuzioni Linux
  - MacOSX, Freebsd e... Windows!
  - Ormai presente su tutte le piattaforme Cloud (AWS, Heroku, OpenShift)
- Clients semplici e completi
  - Psql (riga di comando) e Pgadmin (grafico)
- Aderenza agli standard ANSI SQL 2008 e ACID
- Configurazione pronta all'uso
- Licenza

Semplicità(installazione)

Esempio di installazione su Debian

```
$ sudo su -
$ vi /etc/apt/sources.list.d/pgdg.list

# Inserire questa riga nel file
deb http://apt.postgresql.org/pub/repos/apt/ wheezy-pgdg main

$ curl https://www.postgresql.org/media/keys/ACCC4CF8.asc |apt-key add -

$ apt-get update
$ apt-get install postgresql-9.4
$ su -l postgres
$ psql -U postgres template1
psql (9.4.2)
Digita "help" per avere un aiuto.

template1=# CREATE DATABASE postgresql_it;
CREATE DATABASE

template1=# \c postgresql_it
```

Semplicità(psql)

```
postgresql_it=# \i crea_tabella_prodotto.sql
CREATE TABLE

postgresql_it=# \dt (mostra le tabelle definite)
postgresql_it=# SELECT * FROM prodotto; (tab completion)
postgresql_it=# \d prodotto (mostra la definizione della tabella)
postgresql_it=# \x (abilita l'output espanso)
postgresql_it=# \set marca 'Samsung'  (imposta una variabile)
postgresql_it=# SELECT * FROM prodotto WHERE marca = :'marca'
postgresql_it=# \help SELECT (help sui comandi SQL)
```

```
mysql> \. crea_tabella_prodotto.sql
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW TABLES (mostra le tabelle definite)
mysql> SELECT * FROM prodotto (tab completion)
mysql> SHOW COLUMNS FROM prodotto (mostra la definizione della tabella)
mysql> SELECT * FROM prodotto\G (abilita l'output espanso)
mysql> SET @marca='Samsung';  (imposta una variabile)
mysql> SELECT * FROM prodotto WHERE marca = @marca;
mysql> HELP SELECT
```


Semplicità(licenza)

• PostgreSQL

PostgreSQL is released under the PostgreSQL License, a liberal Open Source license, similar to the BSD or MIT licenses.

PostgreSQL Database Management System (formerly known as Postgres, then as Postgres95)

Portions Copyright (c) 1996-2014, The PostgreSQL Global Development Group

Portions Copyright (c) 1994, The Regents of the University of California

Permission to use, copy, modify, and distribute this software and its documentation for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies.

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

• MySQL

•http://www.gnu.org/licenses/gpl-2.0.html

•http://www.mysql.com/about/legal/licensing/foss-exception/

•http://www.mysql.com/about/legal/licensing/oem/

...Riassunto...

•MySQL can be used without cost if an application is locally developed and not used commercially. It is only when the resulting solution is to be sold to customers that the question of licensing comes into play. This rule is expressed on the MySQL home page as follows: **Free use for those who never copy, modify or distribute.**

•MySQL can be used freely within a web site. If you also develop a PHP application and install it with your Internet service provider, you do not have to make your PHP code freely available in the sense of GPL.

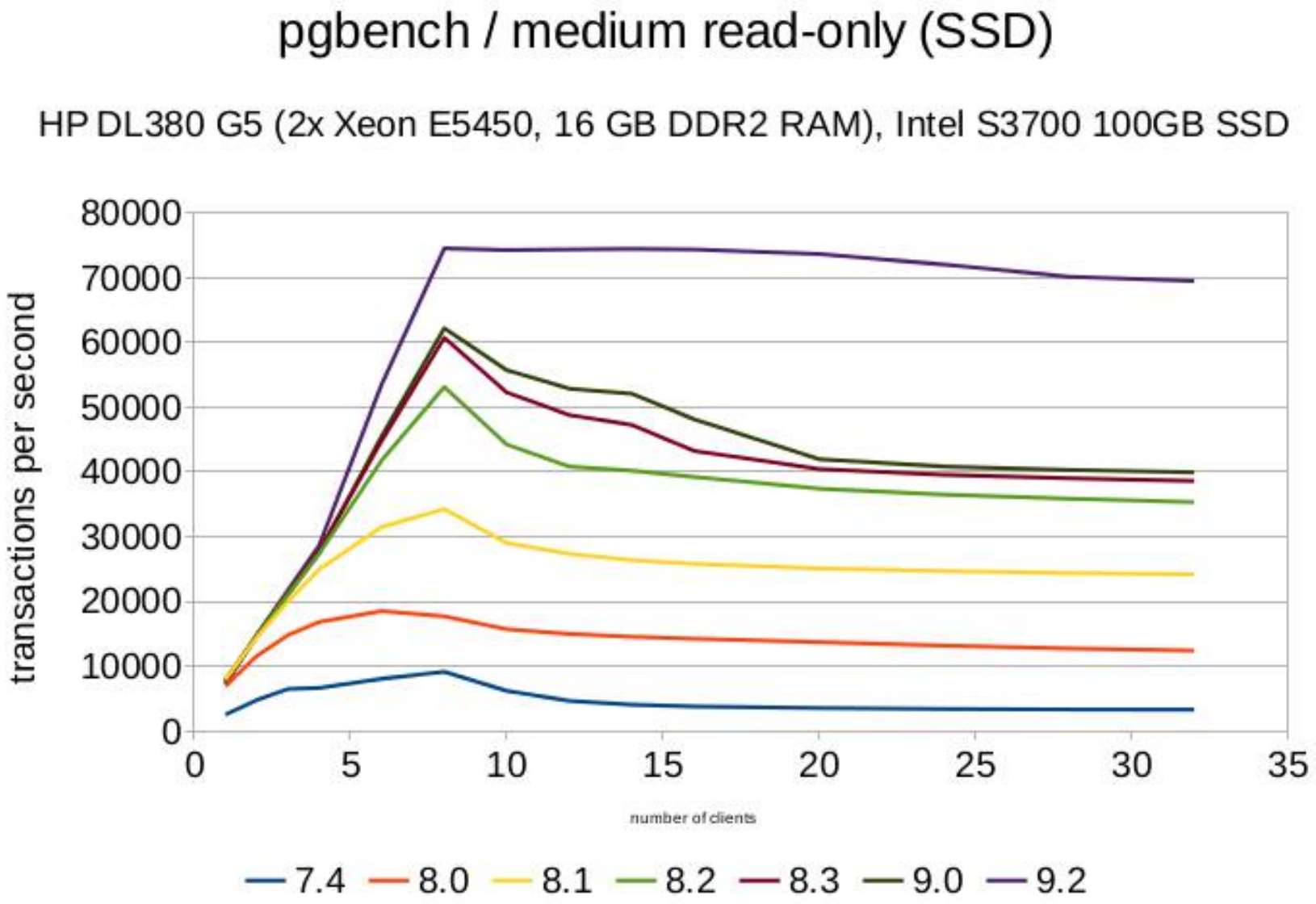
•Likewise, an Internet service provider may make MySQL available to its customers without having to pay MySQL license fees. (Since MySQL is running exclusively on the ISP computer, this application is considered internal.)

•Finally, MySQL license can be used free of charge for all projects that themselves run under the GPL or comparable free license.

Scalabilità(performance)

- Scalabilità orizzontale
  - Data partitioning: partizionamento di tabelle nello stesso DB
  - Data sharding: partizionamento su server DB diversi
- Table spaces
  - Possibilità di memorizzare le tabelle più utilizzate su dischi/storage più performanti
- Hot standby(s)
  - Distribuzione query in sola lettura su più server DB
- Viste materializzate

Scalabilità(performance)



http://blog.pgaddict.com/posts/performance-since-postgresql-7-4-to-9-4-pgbench

Scalabilità(applicazioni)

- Transazioni:
  - anche su DDL (CREATE/DROP TABLE, ALTER TABLE, ...)
  - Savepoints (punto di ripristino all'interno di una transazione)
- Tipi di dato:
  - XML, JSON, Range, Array, Tipi geometrici, Tipi composti
- Query:
  - Window functions, CTE
- Estensioni:
  - Nuove funzionalità con un solo comando: http://pgxn.org/

Scalabilità(applicazioni)

- Motore Full Text Search
- Linguaggi procedurali:
  - PL/pgSQL, PL/Perl, PL/Python, C
- Table inheritance
- Subquery
- Espressioni regolari nelle query
- Schemi
- Triggers
- Integrità referenziale

Sicurezza(accesso ai dati)

- Gestione avanzata dei permessi utente
  - Utenti e gruppi di utenti (ruoli)
  - Permessi su singola colonna di una tabella

- Accesso al database:
  - Gestito dal file pg_hba.conf
  - Possibilità di limitare per indirizzo IP
  - Svariate tipologie di accesso (password, ldap, etc)
  - Possibilità di vincolare l'accesso via SSL

Sicurezza(backup)

PostgreSQL offre almeno tre soluzioni di backup:

- **Backup Logico**: il classico dump SQL, pg_dump e pg_restore
  - Disaster recovery
  - Aggiornamento a nuova major release
- **Backup Fisico**: copia fisica del db con i log transazionali
  - Point in time recovery
- **Replica**: Master-Slave, anche in cascata
  - Riduzione dei tempi di ripristino in caso di crash

Barman: soluzione opensource per la gestione dei backup

Pg(SQL) = S^3 : un esempio

DDL = Data Definition Language (comandi CREATE e DROP TABLE, etc)

PostgreSQL

```
db=# BEGIN;
BEGIN

db=# DROP TABLE prodotto;
DROP TABLE;

db=# ROLLBACK;
ROLLBACK

db=# \dt
```

MySQL

```
mysql> BEGIN;
Query OK, 0 rows affected (0,00 sec)

mysql> DROP TABLE prodotto;
Query OK, 0 rows affected (0,01 sec)

mysql> ROLLBACK;
Query OK, 0 rows affected (0,01 sec)

mysql> SHOW TABLES;
```

Che risultato vi aspettate?

```
              Lista delle relazioni
 Schema |      Nome    |  Tipo   | Proprietario
--------+--------------+---------+-----------
 public |   prodotto   | tabella | postgres
```

```
Empty set (0,00 sec)
```

...ma non lo usa Nessuno!

Ecco qualche esempio di "**Nessuno**"!

Courtesy of Gabriele Bartolini Keynote PgDay 2014

Ed in caso di Problemi? A chi mi rivolgo?

- **Comunità internazionale**
  - Mailing list: http://www.postgresql.org/list/
  - Canale IRC: http://www.postgresql.org/community/irc/
- In Italia esiste **IT-PUG**, fondato nel 2007
  - http://www.itpug.org
  - Organizza il PG-DAY Italiano
  - Mailing List in Italiano
- Società o **Professionisti** esperti

NoSQL!

- Cosa vuol dire?
  - Not Only SQL
  - Alcuni casi d'uso:
    - Basi dati orientate al documento
    - Database a grafo
    - Chiave/Valore

- L'interfacciamento con alcuni DB NoSQL avviene via API REST HTTP e JSON come payload

- PostgreSQL ha introdotto il tipo di dato JSON
  - Da 9.2 come estensione e nativamente da 9.3
  - Da 9.4, tipo JSONB: più veloce e con operatori aggiuntivi

The NoSQL Way

– La “tabella” prodotto in un db NoSQL (ad esempio MongoDB):

```
{
    "id": 1,
    "nome": 'IPhone 6",
    "prezzo": 700.00,
    "quantita": 5
}
```

– E se volessi aggiungere un campo con le caratteristiche?

```
{
    "id": 1,
    "nome": "IPhone 6",
    "prezzo": 700.00,
    "quantita": 5,
    "caratteristiche": {
        "memoria": "16GB",
        "schermo": "4.7",
        "contrasto": "1400:1"
    }
}
```

The SQL Way

```
– La tabella prodotto in un db SQL:

CREATE TABLE prodotto (
    id SERIAL PRIMARY KEY,
    nome TEXT,
    marca TEXT,
    prezzo DECIMAL(10,2),
    quantita INTEGER
);

– E se volessi aggiungere le caratteristiche al prodotto?

CREATE TABLE caratteristica(
    id SERIAL PRIMARY KEY,
    nome TEXT NOT NULL
);

CREATE TABLE caratteristica_prodotto (
    id_prodotto INTEGER NOT NULL REFERENCES prodotto(id),
    id_caratteristica INTEGER NOT NULL REFERENCES caratteristica(id),
    valore TEXT NOT NULL,
    PRIMARY KEY(id_prodotto, id_caratteristica)
);
```

An (available) PostgreSQL Way

```
– La tabella prodotto in PostgreSQL si può definire come una tabella SQL
standard

– E se volessi aggiungere le caratteristiche al prodotto?

ALTER TABLE prodotto ADD COLUMN caratteristiche JSONB;

INSERT INTO prodotto (nome, marca, prezzo, quantita, caratteristiche) VALUES (
    'IPhone 7',
    'Apple',
    2700.00,
    6,
    '{
       "memoria": "1TB",
       "schermo": "9.7",
       "contrasto": "5000:1"
    }'
);

SELECT * FROM prodotto WHERE
    marca = 'Apple' AND
    caratteristiche->>'memoria' = '1TB';
```

Window Functions

- Una window function è simile ad una funzione di raggruppamento
  - Effettua dei calcoli su un gruppo di righe (la cosiddetta finestra)
  - Non fa collassare le righe in un'unica riga
- La "finestra" viene specificata con le istruzioni "OVER" e "PARTITION BY"
- Sono disponibili tutte le funzioni di aggregazione più alcune aggiuntive:
  - row_number()
  - rank()
  - first_value()
  - last_value()
- http://www.postgresql.org/docs/9.4/interactive/functions-window.html

Window Functions

id	nome	dipartimento	salario
1	JOHNSON	ADMIN	18000.00
2	HARDING	MANAGER	52000.00
3	TAFT	SALES	25000.00
4	HOOVER	SALES	27000.00
5	LINCOLN	TECH	22500.00
6	GARFIELD	MANAGER	54000.00
7	POLK	TECH	25000.00
8	GRANT	ENGINEER	32000.00
9	JACKSON	CEO	75000.00
10	FILLMORE	MANAGER	56000.00
11	ADAMS	ENGINEER	34000.00
12	WASHINGTON	ADMIN	18000.00
13	MONROE	ENGINEER	30000.00
14	ROOSEVELT	CPA	35000.00

Window Functions

```
– confrontare lo stipendio di un dipendente rispetto alla media
– del suo dipartimento

SELECT nome, dipartimento, salario,
       avg(salario) OVER (PARTITION BY dipartimento)
FROM dipendente;
```

nome	dipartimento	salario	avg
JOHNSON	ADMIN	18000.00	18000.00
WASHINGTON	ADMIN	18000.00	18000.00
JACKSON	CEO	75000.00	75000.00
ROOSEVELT	CPA	35000.00	35000.00
GRANT	ENGINEER	32000.00	32000.00
ADAMS	ENGINEER	34000.00	32000.00
MONROE	ENGINEER	30000.00	32000.00
HARDING	MANAGER	52000.00	54000.00
GARFIELD	MANAGER	54000.00	54000.00
FILLMORE	MANAGER	56000.00	54000.00
HOOVER	SALES	27000.00	26000.00
TAFT	SALES	25000.00	26000.00
POLK	TECH	25000.00	23750.00
LINCOLN	TECH	22500.00	23750.00

Window Functions

```
– ordinare i dipendenti per stipendio descrescente nello stesso dipartimento

SELECT nome, dipartimento, salario,
       rank() OVER (PARTITION BY dipartimento ORDER BY salario DESC)
FROM dipendente;

    nome     | dipartimento | salario  | rank
-------------+--------------+----------+------
 JOHNSON     | ADMIN        | 18000.00 |    1
 WASHINGTON  | ADMIN        | 18000.00 |    1
 JACKSON     | CEO          | 75000.00 |    1
 ROOSEVELT   | CPA          | 35000.00 |    1
 ADAMS       | ENGINEER     | 34000.00 |    1
 GRANT       | ENGINEER     | 32000.00 |    2
 MONROE      | ENGINEER     | 30000.00 |    3
 FILLMORE    | MANAGER      | 56000.00 |    1
 GARFIELD    | MANAGER      | 54000.00 |    2
 HARDING     | MANAGER      | 52000.00 |    3
 HOOVER      | SALES        | 27000.00 |    1
 TAFT        | SALES        | 25000.00 |    2
 POLK        | TECH         | 25000.00 |    1
 LINCOLN     | TECH         | 22500.00 |    2
(14 righe)
```

Foreign Data Wrappers

- Sono estensioni che consentono di collegare Postgres ad altri database o sorgenti dati
- https://wiki.postgresql.org/wiki/Foreign_data_wrappers
- Esempio: file data wrapper

```
$ apt-get install postgresql-contrib-9.4

CREATE EXTENSION file_fdw;
CREATE SERVER filesystem_server FOREIGN DATA WRAPPER file_fdw;

CREATE FOREIGN TABLE acquisto (
   data_ora timestamp NOT NULL,
   cassa text NOT NULL,
   id_prodotto integer NOT NULL
) SERVER filesystem_server
OPTIONS
(format 'text', filename '/vagrant/acquisti.csv', delimiter E'\t', null '');
```


Viste Materializzate

- Le "Viste" in PostgreSQL sono una semplice riscrittura di una query

```
CREATE VIEW v_prodotto AS SELECT * FROM prodotto WHERE marca='Samsung';

EXPLAIN SELECT * FROM v_prodotto;

                          QUERY PLAN
---------------------------------------------------------------
 Seq Scan on prodotto  (cost=0.00..18.62 rows=3 width=88)
   Filter: (marca = 'Samsung'::text)
```

- Le "Viste materializzate" memorizzano i dati in una tabella:
  - Query di selezione molto più veloci
  - Non sono consentite operazioni di INSERT, DELETE, UPDATE
  - Possono essere aggiornate globalmente

Viste Materializzate

```
CREATE MATERIALIZED VIEW acquisto_materializzato
   AS SELECT * FROM acquisto;

EXPLAIN SELECT * FROM acquisto_materializzato;
                          QUERY PLAN
-------------------------------------------------------------
 Seq Scan on acquisto_materializzato  (cost=0.00..21.00 rows=1100 width=44)

– Un insert nella tabella originale non viene “visto” nella vista
materializzata

– Bisogna usare lo statemente “REFRESH”
REFRESH MATERIALIZED VIEW acquisto_materializzato;


SELECT
   (SELECT COUNT(*) FROM acquisto_materializzato)  AS acquisti_mater,
   (SELECT COUNT(*) FROM acquisto) AS acquisti;

\watch 5
```

Viste Materializzate e Foreign Data Wrappers

- Le viste materializzate consento di velocizzare moltissimo l'accesso a sorgenti dati esterne:
  - Importano i dati nativamente in PostgreSQL
  - Le viste materializzate sono indicizzabili
- Da PostgreSQL 9.4, le viste possono essere aggiornate senza ottenere un LOCK esclusivo sulla vista materializzata

Vediamo i prezzi in Amazon... da PostgreSQL!

- Usiamo in modo fantasioso:
  - Foreign data wrapper www_fdw
  - Stored Procedures PL/PGSQL
  - Json

- Sarebbe bello se potessimo fare un confronto tra il nostro prezzo ed il prezzo a seguito di una ricerca su Amazon

Fantascienza? No, con PostgreSQL si può!

```
CREATE EXTENSION www_fdw;

CREATE SERVER chiedi_ad_amazon FOREIGN DATA WRAPPER www_fdw
    OPTIONS ( uri 'https://www.kimonolabs.com/api/7cer02oo?&apikey=zJZoDC
hUTERCpbMVGbWjw4JBvugQ1VeZ&kimmodify=1',
              response_deserialize_callback 'interpreta_ricerca'
            );

CREATE USER MAPPING FOR postgres SERVER chiedi_ad_amazon;

CREATE FOREIGN TABLE amazon (
  id integer,
  description text,
  pricing text
) SERVER chiedi_ad_amazon;


CREATE OR REPLACE FUNCTION interpreta_ricerca(options WWWFdwOptions,
response text)
    RETURNS SETOF amazon AS $$
...
```

Et voilà!

```
WITH amazon AS (SELECT * FROM amazon)
   SELECT id, nome, description AS amazon_desc, prezzo, pricing AS prezzo_amazon
   FROM prodotto INNER JOIN amazon USING(id) WHERE id=1;

-[ RECORD 1 ]-+---------------------------
id            | 1
nome          | IPhone 6
amazon_desc   | Apple iPhone 6 Plus 16GB 4G Grigio
prezzo        | 700.00
prezzo_amazon | EUR 715,00
-[ RECORD 2 ]-+---------------------------
id            | 1
nome          | IPhone 6
amazon_desc   | Apple iPhone 6 Plus 16GB Argento MGA92QL/A
prezzo        | 700.00
prezzo_amazon | EUR 743,83
-[ RECORD 3 ]-+---------------------------
id            | 1
nome          | IPhone 6
amazon_desc   | Apple iPhone 6 16GB 4G Grigio
prezzo        | 700.00
prezzo_amazon | EUR 654,99
...
```

Grazie!

denis@gasparin.net

http://www.gasparin.net