SQL - Intro

Machine Learning Journal Club - Unito

SQL - 1

Structured **Q**uery **L**anguage

- Linguaggio per interfacciarsi con i database
- Permette di effettuare operazioni sui dati (inserimento, retrieve, modifica, ecc..)

SQL - 2

SQL è solo un protocollo per comunicare con i DB. Possiamo utilizzarlo in (quasi) tutti gli ambienti e linguaggi.

DB

- Cos'è un database?

DB - 2

- Per ora ci basta immaginarlo come una grossa tabella, in cui possiamo memorizzare i dati che ci servono ($\approx$ spreadsheet)

DB - Esempio

tabella: fruit

name	**price**	**quantity**
bananas	1.10	20
oranges	1.05	15
apples	0.90	30
tomatoes	1.50	45

Esempio di query:

- Quali prodotti sono in magazzino in quantità minore di 25?
- Ottenere la lista di prodotti in base a prezzo/quantità crescente/decrescente

DB - Creazione tabella

Creiamo la tabella con il costrutto CREATE TABLE;
successivamente la popoleremo con il costrutto INSERT.

- Nei DB ogni tabella ha bisogno di una (o più) colonne indice, che svolgano la funzione di chiave per identificare univocamente un'entry (riga). Nel nostro caso possiamo usare il nome del prodotto come key.
- Molto spesso è usato un id numerico, generato in automatico dal DB

SQL - CREATE TABLE

Forma generale:

```
CREATE TABLE table_name (
  column1 datatype [..],
  column2 datatype [..],
  ...
)
```

SQL - CREATE TABLE 2

Query in SQL standard:

```
CREATE TABLE fruit (
  name VARCHAR(20) PRIMARY KEY,
  price FLOAT,
  quantity INTEGER
)
```

Query in SQLite (molto usato in python)

```
CREATE TABLE fruit (
  name TEXT PRIMARY KEY,
  price REAL,
  quantity INTEGER
)
```

(mapping da tipi SQL -> SQLite
https://www.sqlite.org/datatype3.html#affinity_name_examples)

SQL - TRYSQL

Ambiente online per provare query sql:

https://bit.ly/2YXJ8xX

SQL - INSERT

```
INSERT INTO table_name (col1, col2, col3, ..)
VALUES (val1, val2, val3, ..)
```

SQL - INSERT 2

```
INSERT INTO fruit (name, price, quantity)
VALUES ("bananas", 1.10, 20)
```

più inserimenti in un colpo solo:

```
INSERT INTO fruit (name, price, quantity)
VALUES ("oranges", 1.05, 15),
       ("apples", 0.90, 30),
       ..completare..
```

SQL - SELECT 1

Per ottenere i dati contenuti in una tabella:

```
SELECT (col1, col2, ...) from table_name
```

per selezionare la tabella intera:

```
SELECT * from table_name
```

SQL - SELECT 2

Esempio: selezionare nome e prezzo dalla tabella fuit

```
SELECT name, price from fruit
```

SQL - SELECT 3

- ORDER BY - Ordinare i dati:

```
SELECT * from fruit ORDER BY price ASC/DESC
```

- WHERE - filtrare i dati:

```
SELECT * from fruit WHERE quantity > 25
```

SQL - SELECT 4

Si possono specificare più criteri insieme:

```
SELECT * from fruit
WHERE quantity > 25 and price > 1.0
ORDER BY price DESC
```

(order matters)

SQL - UPDATE

Modificare i dati di una tabella:

```
UPDATE table_name
SET col1=val1, col2=val2, ..
WHERE condition
```

SQL - UPDATE 2

```
UPDATE fruit
SET price=1.0
WHERE name="apples"
```

Nel nostro esempio, **name** svolge la funzione di chiave (indice) ed è una stringa.

SQL - KEY

Se avessimo una tabella di utenti/persone, usare il campo *nome* come chiave sarebbe una buona scelta?

SQL - JOIN

Di solito, le informazioni in un database sono distribuite su più tabelle (*best practice: no ridondanza di informazioni*), e per questo è spesso necessario aggregare i dati provenienti da più tabelle.

SQL - JOIN ESEMPIO (I)

customers

customerId	**name**	**surname**	**phone**
101	Elon	Tusk	(302)356780193
102	Bill	Gates	(451)184920981
103	Johnny	Depp	(361)081726389
104	Morgan	Freeman	(039)012398739

products

productId	**productName**	**price**
401	Tesla	30000
402	Rocket	100000
403	Tofu	10
404	Uncle Bob's Organic Dried Pears	12

SQL - JOIN ESEMPIO (II)

orders

orderId	**customerId**	**productId**	**quantity**
201	101	403	159
202	103	402	23
203	102	404	10

SQL - JOIN ESEMPIO (III)

- Come facciamo ad aggregare i dati in modo più semplice da leggere? $\rightarrow$ JOIN

```
SELECT customers.name,
       customers.surname,
       orders.productId,
       orders.quantity
FROM orders
INNER JOIN customers ON
  customers.customerId=orders.customerId
```

Cosa restituirà questa query? *(tip: guardare SELECT)*

SQL - JOIN ESEMPIO (IV)

name	**surname**	**productId**	**quantity**
Elon	Musk	403	159
Johnny	Depp	402	23
Bill	Gates	404	10

- Se vogliamo *productName* dobbiamo effettuare un'altra JOIN con la tabella *products*

SQL - JOIN ESEMPIO (V)

```
SELECT customers.name,
       customers.surname,
       products.name,
       orders.quantity
FROM orders
INNER JOIN customers ON
        customers.customerId=orders.customerId
INNER JOIN products ON
        products.productId=orders.productId
```

SQL - JOIN ESEMPIO (VI)

name	**surname**	**products.name**	**quantity**
Elon	Musk	Tofu	159
Johnny	Depp	Rocket	23
Bill	Gates	Uncle Bob's Organic Dried Pears	10