boolean

CORSO FULL STACK WEB DEVELOPER

DATABASE

Come si interagisce con un DB?

Con il linguaggio SQL!

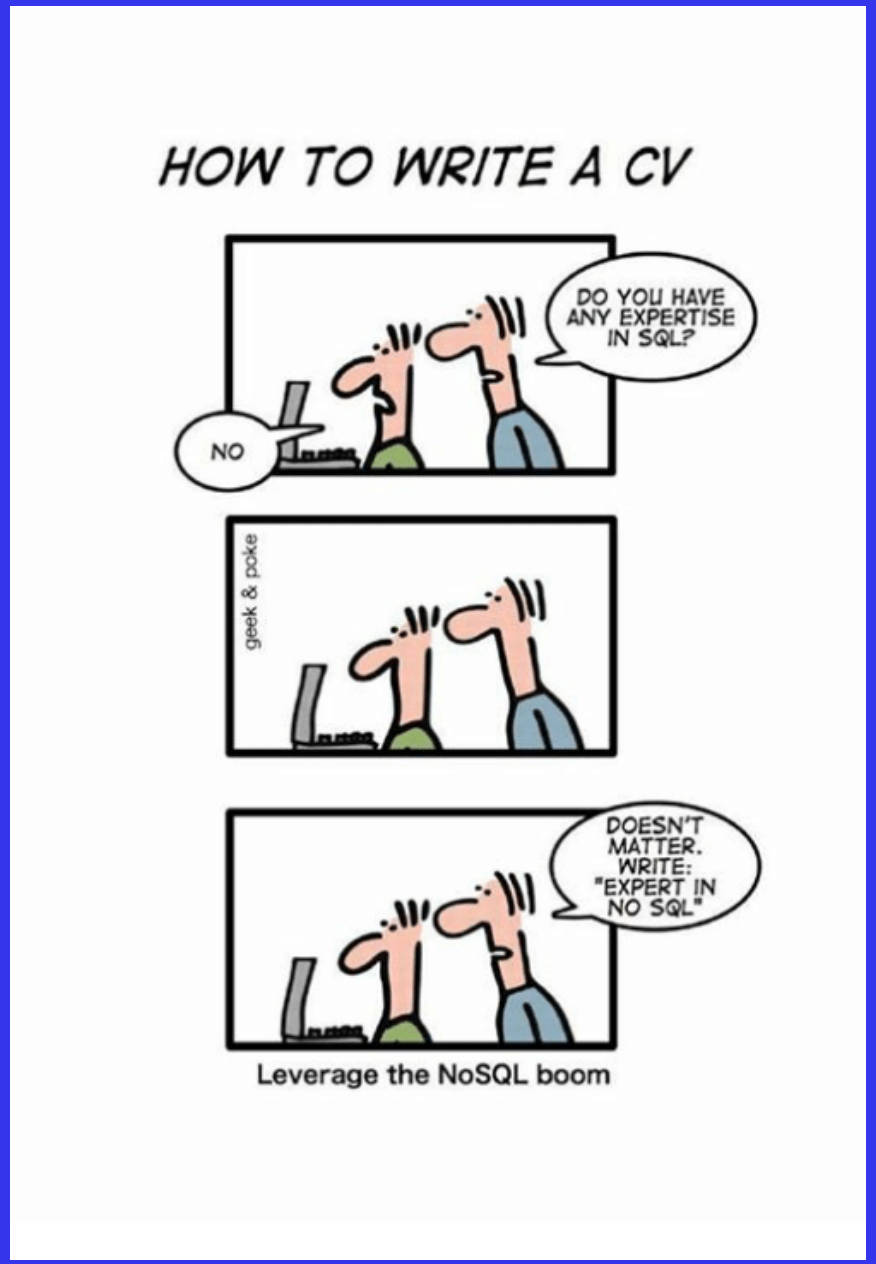
SQL – Structured Query Language

SQL è un linguaggio standard per interagire con i **RDBMS** e offre molteplici funzioni:

- creare e modificare **schemi di database**
- **inserire, modificare e gestire** dati memorizzati
- **interrogare** i dati memorizzati
- creare e gestire strumenti di controllo ed **accesso ai dati**

Nasce nel 1974, negli stessi laboratori IBM dove vengono teorizzati i database relazionali, come linguaggio prevalentemente **dichiarativo**.
L'obiettivo è quello di creare un linguaggio per interrogare i database **semplice e vicino al linguaggio naturale**.


YES I DO HAVE A SPECIAL SKILL SET...
WHAT? NO... WHAT THE HELL IS SQL?
imgflip.com


HOW TO WRITE A CV
DO YOU HAVE ANY EXPERTISE IN SQL?
NO
geek & poke
DOESN'T MATTER. WRITE: "EXPERT IN NO SQL"
Leverage the NoSQL boom

GUI di gestione database

Diamoci una mano

SQL è il linguaggio con cui vengono descritti i comandi al database, ma come vengono inviati?

- Da **riga di comando**: tramite il comando *mysql*
- Utilizzando gli **strumenti** forniti da ogni linguaggio di programmazione
- Tramite **programmi esterni** che forniscono un'interfaccia grafica

Utilizzeremo tutti i metodi, ora installiamo uno dei programmi esterni più famosi: **phpMyAdmin**

LIVE CODING

Mysql Command Line

Come usiamo MySql da linea di comando?

Per avviare la shell mysql, possiamo usare questo comando (l'opzione per indicare il numero della porta è facoltativa):

Windows

```
C:\MAMP\bin\mysql\bin\mysql -u root -p -P 3366
```

Mac

```
/Applications/MAMP/Library/bin/mysql -u root -p -P 3366
```

Subito dopo aver digitato questo comando, il prompt ci chiede la password.

In MAMP, la **password di default** per l'utente root è: **root**

Un database, due database, N database

Un server MySQL può contenere più database

SQL - Create database

Cremiamo un nuovo database

```sql
CREATE DATABASE `nomedatabase`
```

Crea un database con nome `nomedatabase`

```sql
USE `nomedatabase`
```

Usa il database `nomedatabase`

SQL - Create schema

Creiamo lo schema con:
nome colonne, data type, primary key e foreign key

```sql
CREATE TABLE `persone` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(45) NOT NULL,
  `cognome` VARCHAR(45) NOT NULL,
  `data_di_nascita` DATE NULL,
  PRIMARY KEY (`id`)
);
```

```sql
CREATE TABLE `ordini` (
    `id` INT NOT NULL,
    `n_ordine` INT NOT NULL,
    `persona_id` INT,
    PRIMARY KEY (`id`),
    FOREIGN KEY (`persona_id`) REFERENCES persone(`id`)
);
```

SQL - Cambiamento Schema

Cambiamo una tabella una volta che è stata creata

```sql
ALTER TABLE `table_name`
ADD `column_name` datatype;
```

Aggiungere una colonna

```sql
ALTER TABLE `table_name`
DROP COLUMN `column_name`;
```

Eliminare una colonna

```sql
ALTER TABLE `table_name`
MODIFY COLUMN `column_name` datatype;
```

Modificare il datatype

SQL - Cancellazione di una tabella

Cancelliamo ...

```sql
DROP TABLE `table_name`;
```

Cancellare **definitivamente** una tabella e, di conseguenza, tutti i dati che erano in essa presenti!

Query base - lettura

La query base, la più utilizzata, per leggere i record da una tabella.

Selezioniamo tutte le colonne di una tabella

SELECT cosa? **Tutte le colonne ***
FROM da dove? Dalla tabella **movies**

```
1 SELECT *
2 FROM `movies`;
```

Il **;** termina la nostra query.
In questo caso non è essenziale, ma lo diventerebbe nel caso volessimo eseguire più di una query nello stesso comando.

Pillole di Best Practice

Il nostro codice SQL potrebbe essere scritto anche così: **select * from movies**
Tutto su una riga, tutto minuscolo e senza punto e virgola finale.
In realtà, per la miglior lettura del codice, solitamente si cerca di:

1. Scrivere le **keyword MySql in maiuscolo** (SELECT, FROM). Tutto il resto sarà minuscolo
2. **Andare a capo** ad ogni keyword sql
3. **Usare il punto e virgola** ogni qualvolta finisca una query
4. **Usare i backtick** per delimitare i nomi di tabelle e colonne

SQL - Filtrare solo alcune colonne

Filtriamo le colonne che vogliamo vedere.

```sql
SELECT `title`, `original_title`
FROM `movies`;
```

SQL - Gli Operatori

Gli operatori SQL sono simili a quelli di altri linguaggi di programmazione che abbiamo visto fino ad ora, con alcune differenze.

=	Uguaglianza	Non con due o tre =, ma **solo con uno!**
<>	Disuguaglianza	Al posto di !=
< oppure **<=**	Minore o minore e uguale	
> oppure **>=**	Maggiore o maggiore e uguale	

Filtrare i dati - WHERE

Controlliamo se la data di un film è uguale a 01/01/2017

```sql
SELECT `title`, `original_title`
FROM `movies`
WHERE `date` = '2017-01-01';

```

Filtrare i dati - AND/OR

Concatetiamo i filtri!

AND	E	Al posto di &&
OR	O	Al posto di ||

```sql
SELECT `title`, `original_title`
FROM `movies`
WHERE `date` = '2017-01-01'
AND `vote` > 3;
```

Il primo filtro sarà preceduto dal **WHERE**, mentre gli altri saranno preceduti o da **AND** o da **OR** in base al risultato che vogliamo ottenere (intersecare i filtri o unire i filtri).

Filtrare i dati - ORDER BY

I risultati filtrati possono essere mostrati in ordine

1. **ascendente**
2. **discendente**

Se non specifichiamo il tipo di ordinamento (ASC o DESC), MySQL utilizza di **default** l'ordinamento **ascendente**.

SQL Functions

Nella SELECT possiamo inserire delle funzioni
che fanno alcuni calcoli basici:

1. **AVG(**nome_colonna**)** Average/Media
2. **SUM(**nome_colonna**)** Somma
3. **COUNT(**nome_colonna**)** Numero record presenti

SQL Functions

Quanti film sono usciti il primo gennaio 2017?

```sql
SELECT COUNT(`id`)
FROM `movies`
WHERE `date` = '2017-01-01'
```

LIVE CODING

ESERCITAZIONE