Databases 101

https://www.w3schools.com/sql/

I database sono strutture ordinate di dati che vengono interrogati attraverso il linguaggio *SQL*. Le operazioni sui database sono di tipo *CRUD* (creation, reading, updating, deleting).

Il vantaggio principale dell'utilizzo di un database è la *persistenza* dei dati.
altri vantaggi sono:

1. la capacità di immagazzinare grandi quantità di dati;
2. offrire funzionalità per gestione e sviluppo di applicazioni;
3. Garantire una buona velocità di elaborazione;
4. Consentire a più utenti di accedere ai dati;
5. Assicurare l'affidabilità dei dati.

La *Modellazione dei dati* è un processo di progettazione del database attraverso la creazione di *tables* (tabelle). Queste contengono dati contrassegnati da un *tipo* (intero, string boolean, etc). I dati sono salvati all'interno del Database per garantirne l'accesso anche tra esecuzioni diverse.

Ci sono 2 modelli principali:

1. Modello *Entità-Relazione*;
2. Modello a *Oggetti*.

Modello Entità-Relazione

Un'*entità* è qualsiasi cosa sul quale sono raccolte informazioni.
Gli *attributi* sono quegli oggetti che descrivono un'entità.
Le *Relazioni* sono oggetti che collegano le entità di un database.

Tutti gli oggetti devono avere nomi unici e significativi.

Il primo ci permette di usare entità (tabelle ad esempio) e relazioni, che sono i modi con cui le tabelle comunicano tra di loro, è quello che useremo principalmente.

I dati di due tabelle diverse potrebbero essere collegati attraverso delle relazioni. Un attributo di una tabella potrebbe essere segnato come *Primary Key*, una chiave che "indicizza" in maniera unica gli elementi di una tabella, come ad esempio il numero di matricola di studenti universitari.

Possiamo collegare due tabelle utilizzando una *Foreign Key*, che "punta" alla chiave primaria di un'altra tabella.

Sintassi MySQL

Per creare ed entrare in un database:

```
CREATE [database_name];

USE [database_name];
```

Per creare una tabella:

```
CREATE TABLE [table_name] ([nome_attributo] [tipo_attributo], ...);
```

si possono aggiungere delle caratteristiche agli attributi, come:

```
CREATE TABLE [table_name] ([nome_attributo] [tipo_attributo] NOT NULL AUTO_INCREMENT
PRIMARY KEY ...., ...);
```

Per aggiungere una Foreign Key, dopo aver dichiarato gli attributi della tabella:

```
CREATE TABLE [table_name] (attributo1 int,
                           ...
                           ...
                           FOREIGN KEY (attributo1) REFERENCES [table_2](attributo2));
```

Per selezionare dei *record* da una tabella si usa un *SELECT* statement:

```
SELECT [attributo_1],[attributo_2],...,[attributo_n] FROM [table_name];
```

Per ottenere tutti gli attributi di un record si usa *:

```
SELECT * FROM [table_name];
```

L'operatore SELECT è *modulare*, si possono aggiungere, in maniera opzionale ma in questo ordine WHERE, ORDER BY, ASC/DESC: con la sintassi:

```
SELECT * FROM [table_name] WHERE [attributo_1] = [qualcosa] ORDER BY
[attributo_per_cui_ordinare] ASC/DESC;
```

A SELECT si può aggiungere l'istruzione DISTINCT per mostrare i record senza ripetizioni:

```
SELECT DISTINCT [attributo_1],...,[attributo_n] FROM [table_name];
```

Per aggiungere delle entrate in una tabella:

```
INSERT INTO [table_name] ([attributo_1],[attributo_2],...[attributo_n]) VALUES
([valore_attributo1],...[valore_attributo_n]);
```

Gli attributi sono assegnati nell'ordine specificato.
Si possono aggiungere più record per volta aggiungendo un'altra tonda dopo la prima con la stessa sintassi.
Se non vengono inseriti gli attributi di partenza, SQL assumerà che stiano venendo assegnati tutti.

Per aggiornare gli elementi di una tabella usare *UPDATE*:

```
UPDATE [table_name] SET [attributo_da_cambiare] = [qualcosa] WHERE [condizione];
```

IMPORTANTE usare sempre WHERE sia per quest'operazione che per la prossima, pena cancellazione dei dati, senza possibilità di recupero.

L'operatore *DELETE* si usa per eliminare un record da una tabella:

```
DELETE FROM [table_name] WHERE [condizione];
```

Per modificare la struttura di una tabella si usa ALTER:

```
ALTER TABLE [table_name] ADD [column_name] [data_type]; -- per aggiungere una colonna

ALTER TABLE [table_name] DROP [column_name] -- rimuove una colonna
```

L'Operatore *LIKE* è usato in combinazione con WHERE per cercare un pattern specifico.
Spesso si usano i caratteri *%* e *_* che indicano rispettivamente qualunque numero di quel carattere e uno solo:

```
SELECT * FROM [table_name]
WHERE [attributo] LIKE 'a%'; -- Cerca solo gli attributi che iniziano per 'a'
```

l'operatore *ALIAS* rinomina le colonne mostrate:

```
SELECT [attributo1] AS [attributo_nuovo_nome] FROM [table_name];
```

si possono rinominare più colonne per volta:

```
SELECT [attributo_1],...,[attributo_n]
AS [attributo_1_nuovo_nome],...,[attributo_n_nuovo_nome] FROM [table_name];
```

WHERE (condizione)

Gli operatori utilizzabili sono:

Operatore	Esempio
uguale: =	nome = "Giggino"
maggiore: >	età > 15
minore: <	età < 15
maggiore e uguale: >=	anno >= 1990

Operatore	Esempio
minore e uguale: <=	anno <= 1990
diverso: <>	città <> "Livorno"
valori tra: *BETWEEN*	codice BETWEEN 10 AND 20
cercare un pattern: *LIKE*	
cercare diversi possibili valori:*IN*	genere IN ("Horror","Commedia")

Le condizioni di un WHERE possono essere concatenate con gli operatori *AND* e *OR*, ad esempio:

```
SELECT * FROM tabella WHERE nome = "franco" AND cognome "Novaga";

--oppure

SELECT * FROM tabella WHERE nome = "franco" OR id = 1;
```

Le condizioni possono essere negate con l'operatore *NOT*:

```
SELECT * FROM tabella WHERE NOT città = "Livorno";
-- L Livorno
```

JOIN STATEMENT

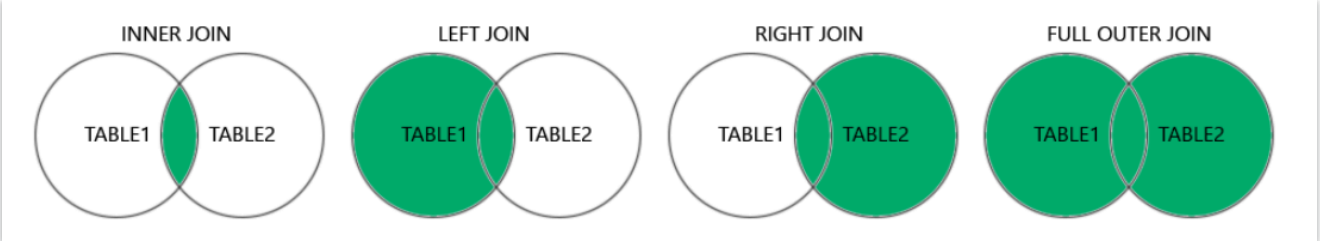
I *JOIN* sono degli statement che permettono di visualizzare i dati di più tabelle contemporaneamente, associando la *PRIMARY KEY* della prima tabella alla *FOREIGN KEY* della seconda.

```
SELECT [table_1_o_2].[attributo_1],...,[table_1_o_2].[attributo_n]
FROM [table_1]
INNER JOIN [table_2] -- INNER JOIN specifica solo gli elementi in comune (AND
insiemistico)
ON [table_1].[FOREIGN_KEY] = [table_2].[PRIMARY_KEY]; -- dove la FK della prima tabella è
uguale alla
-- PK della tabella 2
```

Ci sono vari tipi di JOIN in sql:

1. *INNER JOIN*: seleziona solo gli elementi per cui la condizione *ON* è vera.
2. *LEFT (OUTER) JOIN*: seleziona tutta la tabella di sinistra più elementi per cui la condizione *ON* è vera.
3. *RIGHT (OUTER) JOIN*: lo stesso di LEFT ma con la tabella di destra.
4. *FULL (OUTER) JOIN*: Seleziona tutti gli elementi, da entrambe le tabelle, a patto che ci sia almeno un match.

come mostrato in figura



Altri comandi

Per stampare solo il minimo/massimo di un select:

```
SELECT MIN([attribute]) FROM [table_name];

SELECT MAX([attribute]) FROM [table_name];
```

Per ottenere il numero di apparizioni di un valore, invece:

```
SELECT COUNT([attribute]) FROM [table_name];
```

Per sommare o fare la media tra i valori numerici di una colonna:

```
SELECT SUM([attribute]) FROM [table_name];

SELECT AVG([attribute]) FROM [table_name];
```

sum avg count

GROUP BY raggruppa i risultati in base alle colonne. spesso si usa una funzione aggregata (quelle appena sopra):

```
SELECT COUNT([attribute]) FROM [table_name] -- count è una funzione aggregata
WHERE [condition]
GROUP BY [column_name]
ORDER BY [column_name]; -- potenzialmente il modo di ordinare i record all'interno dei
gruppi
```

HAVING è come un WHERE ma è possibile utilizzare le funzioni aggregate come condizioni a seguito di un GROUP BY:

```
SELECT COUNT(*) FROM [table_name]
GROUP BY [column_name]
HAVING COUNT(*) > 5 -- ad esempio
```

SUB QUERIES

Si possono scrivere queries innestate:

```
-- LA TUA QUERY PREFERITA

WHERE [attributo] = (
-- SUB QUERY
);
```