SQL

SQL è un "linguaggio di programmazione" progettato per manipolare e gestire i dati registrati in un database relazionale.

Un database relazionale è un database che organizza le informazioni in una o più tabelle.

Una tabella è una collezione di dati organizzati in righe e colonne.

Le colonne sono definite come ATTRIBUTI.

Le righe sono definite come RECORD.

Il linguaggio SQL consente all'utente di:
- Definire la struttura delle relazioni del database
- Modificare i dati contenuti nel database, con le operazioni di inserimento, variazione e cancellazione
- Gestire il controllo degli accessi e i permessi per gli utenti
- Porre interrogazioni al database
- Contenere record di grandi quantità

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	auto_increment
Cognome	varchar(100)	NO		NULL	
Nome	varchar(100)	NO		NULL	
Indirizzo	varchar(255)	NO		NULL	
Anno	int(11)	YES		NULL	

Quando si crea una tabella, per ogni attributo va definito il tipo di variabile.
Qui sotto sono i tipi di variabili più utilizzati:

CHAR(size)	Contiene un numero *definito* di caratteri (massimo size)
VARCHAR(size)	Contiene un numero *variabile* di caratteri, massimo (size). Il numero di caratteri massimo è 255, poi viene trasformato in TEXT
TEXT	Contiene una stringa di al massimo 65535 caratteri
MEDIUMTEXT	Contiene una stringa di al massimo 16777215 caratteri
LONGTEXT	Contiene una stringa di al massimo 4294967295 caratteri
INT(size)	Numero intero. Con size (che può essere omesso) si specifica il numero massimo di cifre.
FLOAT (size,d)	Contiene un numero "piccolo" decimale. Size sono le cifre intere, d le cifre decimali.

DOUBLE(size,d)	Come float ma numero più grande
DATE()	Contiene una data YYYY-MM-DD (a partire dall'anno 1000)
DATETIME()	Contiene data e ora YYYY-MM-DD HH:MI:SS
TIMESTAMP()	Contiene data e ora YYYY-MM-DD HH:MI:SS inserite automaticamente.
TIME()	Contiene la data HH:MI:SS
YEAR()	Contiene un anno (YY o YYYY)

TIMESTAMP accetta differenti formati come:

YYYYMMDDHHMISS, YYMMDDHHMISS, YYYYMMDD, or YYMMDD.

Nella creazione della tabella, possono essere inseriti anche dei **CONSTRAINT** in modo da limitare la tipologia di dati inseriti.

Il constraint viene indicato durante la creazione della tabella, quando viene indicato il tipo di dati.

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
);
```

NOT NULL	In modo che la colonna non possa essere vuota
UNIQUE	Si assicura che tutti i valori di una colonna sia differenti
PRIMARY KEY	È una combinazione di NOT NULL e UNIQUE
DEFAULT	Setta un valore di default, quando non viene inserito nulla

Per visualizzare i dati da SQL si usa il comando SELECT

Prima di iniziare un piccolo approfondimento su SQL.
Durante la select può diventare utile fare calcoli, tempistiche, ecc.; per tale ragione SQL include delle funzioni.

- SQL AGGREGATE FUNCTION
- SQL SCALAR FUNCTION

SQL AGGREGATE FUNCTION

Queste funzioni ritornano un dato, calcolato da valori letti dalle colonne attraverso una select.

AVG() - Ritorna la media

```
SELECT AVG(price) FROM magazzino;
```

COUNT() - Conta il numero di righe estratte

```
SELECT count(*) FROM employees WHERE gender = 'f'
```

FIRST() - Ritorna il primo valore

```
SELECT first_name FROM employees  order by first_name ASC limit 1
```

LAST() - Ritorna l'ultimo valore

```
SELECT first_name FROM employees  order by first_name DESC limit 1
```

MAX() - Ritorna il valore massimo

```
SELECT MAX(salary) from salaries;
```

MIN() - Ritorna il valore minimo

```
SELECT MIN(salary) from salaries;
```

SUM() - Ritorna la somma

```
SELECT SUM(salary) from salaries;
```

AS permette di dare un nome al valore calcolato

SQL SCALAR FUNCTION

Queste funzioni ritornano un valore, dato un valore in ingresso singolo

UCASE() - Converte tutto in maiuscolo

LCASE() - Converte tutto in minuscolo

MID() - Estrae dei caratteri da un testo
SELECT MID(first_name,1,1) FROM employees ORDER BY first_name ASC LIMIT 1;

LENGTH() - Ritorna la lunghezza del testo

REVERSE() - Ribalta una stringa

ROUND() - Arrotonda il valore numerico, al numero di decimali specificato

NOW() - Ritorna data e ora corrente

FORMAT() - Formatta come un campo deve essere visualizzato
SELECT FORMAT (2550523.4323, 2);
Riformata con solo due decimali

OPERATORI LOGICI DI COMPARAZIONE

Operator	Description
=	Equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

LEGGERE DAL DB

Per leggere dei campi da una tabella:

```
SELECT colonna1, colonna2…
FROM nome_tabella
```

Per leggere tutto:

```
SELECT * FROM nome_tabella
```

Se prendiamo per esempio un db contenente gli studenti del politecnico e volgiamo estrarre i nomi, ce ne sono molti con lo stesso nome. Vogliamo stampare solamente i nomi, senza ripetizioni.

SELECT DISTINCT viene utilizzato per ritornare solo valori distinti (differenti)

SELECT DISTINCT(first_name) FROM employees

Se si vuole aggiungere una condizione per migliorare la ricerca si usa WHERE

```
SELECT colonna1, colonna2
FROM tabella
WHERE condizione;
```

Per esempio, per visualizzare il nome dei solo clienti italiani

```
SELECT *
FROM clienti
WHERE nazione='italiana'
```

Al contrario, per selezionare i NON italiani

```
SELECT *
FROM clienti
WHERE nazione<>'italiana'
```

Seleziono i campi con nazionalità nulla

```
SELECT *
FROM clienti
WHERE nazione IS NULL;
```

Selezionare i clienti tra l'80 e l'85

```
SELECT *
FROM clienti
WHERE Anno BETWEEN 1980 AND 1985
```

Si possono aggiungere più condizioni concatenandole attraverso AND e OR

Seleziono i clienti italiani tra l'80 e l'85

```
SELECT *
FROM clienti
WHERE nazione = 'italiana'  AND Anno BETWEEN 1980 AND 1985
```

L'operatore IN permette di confrontare in un WHERE più valori

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

Se si vuole confrontare il contenuto di un campo attraverso il where, ma non si conosce l'esatto valore contenuto, o si vuole ricercare determinate parole per iniziale ecc, si può usare il comando LIKE.

Like funziona per mezzo di due wildcards:
% : indica molteplici caratteri
_ : indica un carattere

Cercare nella tabella parole, le parole che contengono or in qualsiasi posizione

```
SELECT parola
FROM parole
WHERE parola LIKE '%or%';
```

Cercare gli utenti che iniziano per a

```
SELECT first_name
FROM customers
WHERE first_name LIKE 'a%'
```

All'esame è accettato anche:

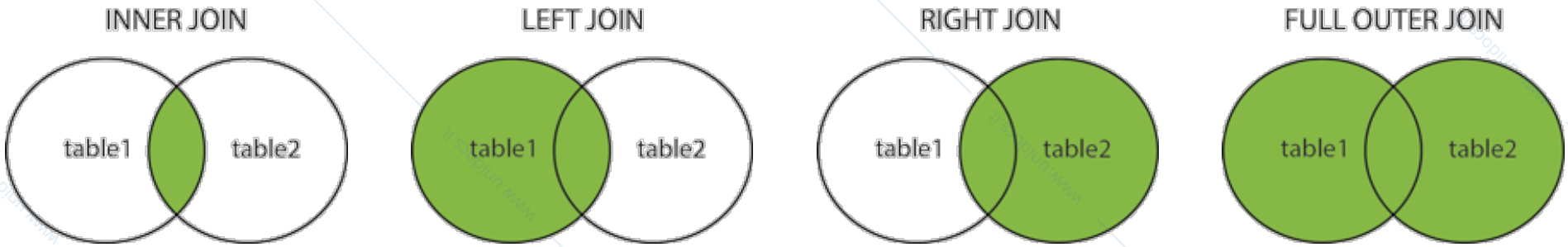
```
SELECT first_name
FROM customers
WHERE first_name = 'a%'
```

JOIN

A volte non basta leggere da una sola tabella, ma occorre unirle per estrarre i dati correnti.

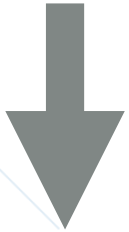
Le unioni funzionano come negli insiemi.

- Si possono unire due tabelle guardandone l'intersezione (solo gli elementi esistenti in entrambe le tabelle risultano nell'unione). INNER JOIN
- Si possono unire inserendo tutte quelle provenienti dalla prima tabella, aggiungendo i dati (ove esistono) per la seconda; LEFT JOIN
- Si possono unire partendo dalla seconda tabella RIGHT JOIN
- Si può unire tutto quanto FULL OUTER JOIN



SQL Alippi prevede la JOIN mediante la condizione logica di uguaglianza applicata dalle chiavi-attributo associate alla relazione.

```
SELECT Customers.CustomerID, CustomerName FROM Customers
JOIN Orders
ON Orders.CustomerID = Customers.CustomerID
WHERE Orders.EmployeeID = 5
```



```
SELECT Customers.CustomerID, CustomerName

FROM Customers, Orders

WHERE Orders.EmployeeID = 5
      AND Orders.CustomerID = Customers.CustomerID
```

JOIN

GROUP BY

Group by è usato con i dati aggregati (COUNT, MAX, MIN, ecc) e permette di calcolare eventuali valori raggruppati per tipo.

ORDER BY

Permette di ordinare il risultato in rodine crescente (ASC) ed in ordine decrescente (DESC).

Es.
Contare i clienti per città e stamparli in ordine decrescente

```
SELECT COUNT(id),  city
FROM Customers
GROUP By city
ORDER BY COUNT(id) DESC;
```

Se si vuole porre delle condizioni sui dati aggregati occorre usare HAVING

Vogliamo visualizzare il numero di clienti per città, solo se >2.

```
SELECT COUNT(id), state_province
FROM customers
GROUP BY state_province
HAVING COUNT(id)>2
ORDER BY COUNT(id) ASC
```

Per testare qualche query SQL:

https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_in

Esercizi:

1. Estrarre dal database, tutte le città dei clienti.

2. Estrarre dal database, tutte le città dei clienti, scrivendole solo una volta

3. Estrarre dal database, tutte le città dei clienti, scrivendole solo una volta, metterle in ordine alfabetico

4. Estrarre dal database, tutte le città dei clienti, scrivendole solo una volta, metterle in ordine alfabetico, chiamando il risultato CITTÀ

5. Estrarre dal database, tutte le città dei clienti, scrivendole solo una volta, metterle in ordine alfabetico, chiamando il risultato CITTÀ. Ordine decrescente.

1.
SELECT City
FROM Customers

2.
SELECT DISTINCT City
FROM Customers

3.
SELECT DISTINCT City
FROM Customers
ORDER BY CITY

4.
SELECT DISTINCT City as Città
FROM Customers
ORDER BY City

5.
SELECT DISTINCT City as Città
FROM Customers
ORDER BY City DESC

Esercizi:

1. Mostrare l'id ordine degli ordini fatti nel1997
2. Mostrare gli ordine del gennaio 1997
3. Contare gli ordine del gennaio 1997
4. Contare gli ordine per ogni mese

Esercizi:

1.
```
SELECT OrderID
FROM Orders
WHERE YEAR(OrderDate) = 1997
```

2.
```
SELECT OrderID
FROM Orders
WHERE YEAR(OrderDate) = 1997 AND MONTH(OrderDate) = 1
```

3.
```
SELECT COUNT(OrderID) AS "ORDINI DI GENNAIO"
FROM Orders
WHERE YEAR(OrderDate) = 1997 AND MONTH(OrderDate) = 1
```

4.
```
SELECT MONTH(OrderDate), COUNT(OrderID) AS “TOT ORDINI"
FROM Orders
WHERE YEAR(OrderDate) = 1997
GROUP BY MONTH(OrderDate)
```

Esercizi:

1. Stampare l'ID del prodotto che ha venduto di più in un unico ordine
2. Stampare in ordine decrescente ID e quantità dei prodotti che hanno venduto di più
3. Stampare l'ID del prodotto che ha venduto di più
4. Stampare il nome e il prezzo del prodotto che ha venduto di più

Esercizi:

1.

```
SELECT ProductID, Quantity FROM OrderDetails
WHERE Quantity = (
                    SELECT MAX(Quantity)
                  FROM OrderDetails
                )
```

2.

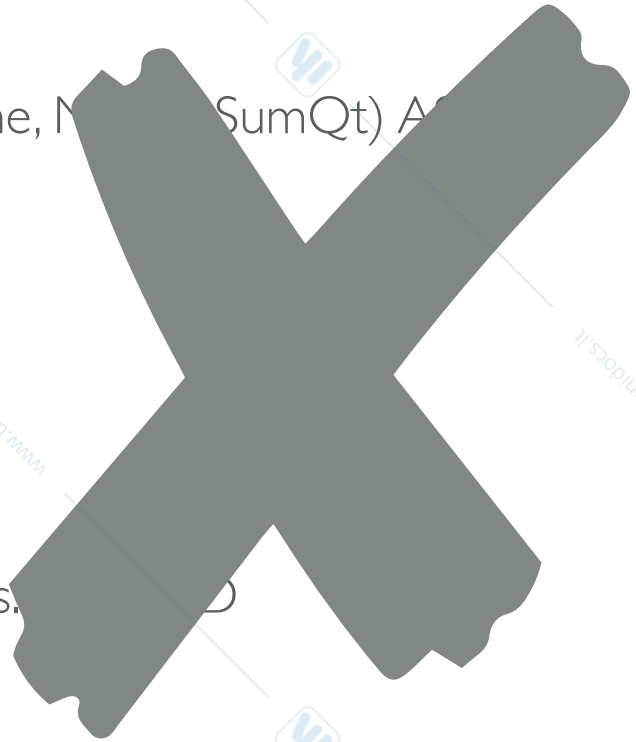
```
SELECT ProductID, SUM(Quantity)
FROM OrderDetails
GROUP BY (ProductID)
ORDER BY SUM(Quantity) DESC
```

3.

```
SELECT ProductID, MAX(s.SumQt) AS VENDITA_MASSIMA
FROM (
        SELECT ProductID pid, SUM(Quantity) SumQt
        FROM OrderDetails
        GROUP BY ProductID) s,

      OrderDetails

WHERE ProductID = s.pid
```

4.

```
SELECT OrderDetails.ProductID, Products.ProductName, M      SumQt) A
VENDITA_MASSIMA
FROM (
        SELECT ProductID pid, SUM(Quantity) SumQt
        FROM OrderDetails
        GROUP BY ProductID) s,

       OrderDetails
JOIN Products ON OrderDetails.ProductID = Products.      D
WHERE OrderDetails.ProductID = s.pid
```



4.

```
SELECT OrderDetails.ProductID, Products.ProductName, MAX(s.SumQt)
FROM (
        SELECT ProductID pid, SUM(Quantity) SumQt
        FROM OrderDetails
        GROUP BY ProductID) s,

        OrderDetails,
        Products,
WHERE OrderDetails.ProductID = s.pid AND OrderDetails.ProductID = Products.ProductID
```

Stampare quante volte ho venduto il prodotto più costoso

1. Trovo il prodotto più costoso
2. Cerco quante volte l'ho venduto

```
SELECT ProductID, ProductName, Price
FROM Products
WHERE Products.Price = (
                        SELECT MAX(Price)
                    FROM Products
                    )
```

```
SELECT COUNT(*) AS "VENDITE PRODOTTO PIU' COSTOSO"
FROM (SELECT ProductID, ProductName, Price
FROM Products
WHERE Products.Price = (
                        SELECT MAX(Price)
                        FROM Products
                      )
      ) ProdottoCostoso,
        OrderDetails
WHERE OrderDetails.ProductID = ProdottoCostoso.ProductID
```

Vogliamo controllare se è tutto vero:

```
SELECT COUNT(*)
FROM OrderDetails
WHERE ProductID = 38
```

1. Selezionare i clienti che sono stati serviti dall'impiegato 5

```
SELECT Customers.CustomerID, CustomerN
FROM Customers
JOIN Orders
ON Orders.CustomerID = Customers.Custom
WHERE Orders.EmployeeID = 5
ORDER BY Customers.CustomerID
```

```
SELECT Customers.CustomerID, CustomerName
FROM Customers, Orders
WHERE Orders.EmployeeID = 5
        AND Orders.CustomerID = Customers.CustomerID
ORDER BY Customers.CustomerID
```

1. Selezionare i clienti che sono stati serviti dall'impiegato Davolli

```
SELECT DISTINCT Customers.CustomerID, CustomerName, Emp
FROM Customers
JOIN Orders
ON Orders.CustomerID = Customers.CustomerID
JOIN Employees
ON Orders.EmployeeID = Employees.EmployeeID
WHERE Employees.LastName = "Davollio"
ORDER BY Customers.CustomerID
```

```
SELECT DISTINCT Customers.CustomerID, CustomerName, Employees.LastName
FROM Customers, Orders, Employees
WHERE Orders.CustomerID = Customers.CustomerID
        AND Orders.EmployeeID = Employees.EmployeeID
        AND  Employees.LastName = "Davollio"
ORDER BY Customers.CustomerID
```