SQL 04/09/2024-05/09/2024

SQL linguaggio per database relazionale

useremo un DBMS, software per gestire database. Useremo MAMP

MAMP: ambente server locale gratuito, esegue MySQL

MySQL: rdbms, software libero, conforme agli standard ANSI SQL, ODBC SQL; compatibile con python, moldi “dialetti”

linguaggio standard per gestire database: tabelle di dati

l’interrogazione al databes avviene tramite QUERY: epressioni/richieste

```
SELECT * FROM world.country; /*importa tutto*/
SELECT Code, Name /* importa colonne Code e Name*/
FROM world.country;
UPDATE
DELETE
INSERT INTO
CREATE DATABASE
ALTER DATABASE
CREATE INDEX
DROP INDEX

CREATE TABLE Students(
        StudentID int PRIMARY KEY,
        FisrstName VARCHAR(50), /*stringa di 50 caratteri*/
        LastName VARCHAR(50)
        );
```

Sintassi molto specifica, non è case sensitive per i comandi

Alcuni sistemi dopo query vogliamo ; per inserire più istruzioni

Schema RELAZIONALE: struttura e relazioni tra tabelle

CHIAVI PRIMARIE: attributo/i che identifica univocamente le righe di una tabella; può avere uno o più campi

CHIAVE ESTERNA: permette relazione tra tabelle

```
CREATE TABLE Enrolmentss(
        EnrolmentID int PRIMARY KEY,
        Name VARCHAR(50), /*stringa di 50 caratteri*/
        FOREIGN KEY (StudentiID)
        FOREIGN KEY (CorsiID)
        );
        -- questo è un commento
```

Operazioni solo per AMMINISTRATORI

```
CREATE DATABASE database_name
        -- questo è un commento
```

SQL non è dinamico, bisogna sempre specificare i TIPI di dato

```
CREATE TABLE Persona(
    PersonID int PRIMARY KEY
    Name varchar(224)
    );

CREATE new_table
    SELECT PersonID, column2
    from Persona
```

PRINCIPALI TIPI IN SQL:

```
CHAR(size) --stringa di grandezza fissa(max 225)
VARCHAR(size) --stringa di grandezza variabile(max 65535)
MEDIUMTEXT -- fino a 16 777 215 caratteri
LONGTEXT -- fino a 4 294 967 295 caratteri
ENUM(val1, val2,...) --lista di valori
BOOL -- 0 e 1
INT
FLOAT
DATE --data in formato YYYY-MM-DD
```

Eliminare tabella:

```
DROP TABLE nome_tabella --elimina la tabella
TRUNCATE TABLE nome_tabella -- elimina i dati ma non la strut
ALTER TABLE nome_tabella -- modifica struttura
ADD nome_colonna datatype -- aggiunge colonna
```

Possiamo avere VINCOLI

```
CREATE TABLE tabella(
        column1 datatype constraint,
        ...);

--Vincoli più comuni
NOT NULL --mai valori nulli
UNIQUE --dati univoci
PRIMARY (FOREIGN) KEY
CHECK --condizione
DEFAULT -- VALORE DI DEFAULT
CREATE INDEX -- crea un indice per recuperare i dati
AUTO_INCREMENT --aumento automatico

--esempi
CREATE TABLE Persona(
        ID int NOT NULL,
        Age int
        ...);

--si possono aggiungere i vincoli dopo
ALTER TABLE Persona
MODIFY Age int NOT NULL;
CONSTRAINT ctn_name PRIMARY KEY UNIQUE (ID)
```

Foreign key = riferimenti ad altre tabelle, permette le relazioni (tabelle figlie)

Istruzione SELECT:

```
SELECT * FROM table_name; --prende tutte le colonne
SELECT column1, column2 FROM table_name;
```

```
SELECT DISTINCT column1
from table_name; --solo elementi univoci

-- filtro condizione
SELECT column1
from table_name
where condition;
--esempio
SELECT * FROM world.country
WHERE Region = 'Antartica';

SELECT name, SurfaceArea FROM world.country
WHERE SurfaceArea > 193;
```

Condizioni con il WHERE:

```
= --uguale
>
<
!= oppure <> --diverso
BETWEEN 0 and 100 --valori compresi
LIKE 's%' --valori che iniziano per s, ci sono altri pattern
IN(0,100,1000,10000) --valori nell'elenco
--null
WHERE column_name IS NULL
WHERE column_name IS NOT NULL
--operatori logici come python
WHERE condition1 AND condition2;
WHERE condition1 AND (condition2 OR NOT condition3)
```

Order BY:

```
select colonna1
from tabella1
order by column1 ASC|DESC;

SELECT * FROM wordl.country
ORDER BY column1 ASC
```

```
SELECT * FROM wordl.country
WHERE condition
GROUP BY column_name(s)

SELECT Country, COUNT(CustomerID)
FROM Customers
group BY Country
ORDER BY COUNT(CustomerID) DESC
```

Group by: raggruppo per lo stesso stato e poi posso contare quanti elementi in quello stato

INSERT INTO: inserire i dati

```
INSERT INTO table_name(column1, column2)
VALUES(value1, value2);
--inserisco in table_name, value1 in column1 e value2 in colu
INSERT INTO table_name
VALUES(value1, value2); --devo mettere però tutti e in ordine
```

UPDATE:

```
UPDATE mome_tabella
SET column1 = value1, column2 = value2
WHERE condition

UPDATE Customers
SET ContactName = 'AlfredSchmidt', City = 'Frankfurt'
WHERE CustomerID = 1
```

DELETE

```
DELETE FROM nome_tabella
WHERE condition

DELETE FROM Customers
WHERE CustomerName = 'Mario Rossi'
```

SELECT TOP (LIMIT)

```
SELECT column_name
FROM table_name
WHERE condition
LIMIT number --numero limitato di record

SELECT * FROM Customers
LIMiT 50 --solo le prime 50 righe trovate
```

ALIAS

Nome temporaneo ad un elemento

```
SELECT column_name AS alias_column_name
FROM table_name AS alias_table_name;
```

ESERCIZIO 1:

Scrivete una query SQL che restituisca solo i record dalla tabella "products"con un prezzo superiore a 50.

```
SELECT productName
FROM `classicmodels.products
WHERE buyPrice > 50
```

ESERCIZIO 2:

Scrivete una query SQL che restituisca tutti i record dalla tabella "orders"ordinati per data in ordine decrescente

```
SELECT *
FROM classicmodels.orders
ORDER BY orderDate DESC
```

Functions:

```
SELECT MIN(column_name)
from table_name
where confition;
```

```
SELECT MAX(PRICE) AS LargestPrice
FROM Products;

SELECT SUM(column_name)|AVG()|
FROM table_name;
```

WILDCARD CHARACTERS: usati con WHERE e LIKE

```
'%' --zero o più caratteri
'bl%'
"_" -- singolo carattere
'h_t' --(hot, hat, hit)
------------------------------
'[]'--ogni carattere nelle quadre
'^' --rifiuta i caratteri indicati
'h[^oa]t'
'-' --range
'c[a-t]t'--tutti i valori compresi

'_r%'-- ha almeno una r come secondo carattere

SELECT * from Customers
WHERE CustomerName LIKE 'a%'

SELECT *
FROM table_name
where column_name IN (SELECT STATEMENT);

SELECT *
FROM Customers
WHERE Country in ('Germany'. 'UK')

SELECT *
FROM Customers
WHERE Country IN (SELECT Country FROM Suppliers)

SELECT *
```

```
FROM Products
WHERE Price BETWEEN 10 AND 20
AND CategoryID NOT IN (!, 2, 3)

SELECT *
FROM Orders
WHERE OrderDate BETWEEN '1996-07-01' and '1996-07-31'
```

ESERCIZIO 3

Scrivete una query SQL che aggiorni il prezzo di tutti i prodotti nella tabella"products" aumentandolo del 10%

```
UPDATE classicmodels.products
SET buyPrice = buyPrice + 0.1*buyprice;
```

ESERCIZIO 4

Scrivete una query SQL che inserisca un nuovo utente nella tabella"customers"

```
INSERT INTO classicmodels.customers(customerNumber,
                                                customerName,
                                                contactLastNa
                        contactFirstName, phone,
                        addressLine1, city, country)
VALUES  (497,'Aldo Corporation',
'Perri', 'Aldo', '3454462120', 'via Mario Rossi',
 'Bologna', 'Italy');
```

ESERCIZIO 5

Scrivete una query SQL che elimini tutti gli ordini nella tabella "orders" con lo stato "Cancelled".

```
DELETE FROM classicmodels.orderdetails
WHERE orderNumber IN (SELECT orderNumber
        FROM classicmodels.orders
        WHERE status = 'Cancelled');
```

```
DELETE FROM classicmodels.orders
WHERE status = 'Cancelled';
```

ESERCIZIO 6

Scrivete una query SQL che restituisca tutti gli utenti dalla tabella"customers" il cui nome inizia con la S e vivono in California.

```
SELECT * FROM classicmodels.customers
WHERE customerName LIKE 's%' and state = 'CA';

SELECT * FROM classicmodels.customers
WHERE contactFirstName LIKE 's%' and state = 'CA';
```

JOIN

interrogare più tabelle insieme

Esempio: selezione colonne da Orders e Customers. Seleziono Orders e faccio inner join con customers, dove specifico (ON) gli elementi che vanno a combaciare

```
SELECT Orders.OrderID, Customers.CustomerName,
                                Orders.OrderDate
FROM Orders
INNER JOIN Customers
ON Orders.CustomerID = Customers.CustomerID
```

- INNER JOIN: valori comuni, intersezione tra le tabelle di cui facciamo il join

```
Select col1
from tab1 --sinistra
INNER JOIN tab2 --destra
ON tab1.col1 = tab2.col1
--intersezione tra le 2 tab con stessa colonna
```

- LEFT/RIGHT (OUTER) JOIN: prendo anche i valori non comuni nella tabella di sinistra/destra

```
SELECT column_name(s)
FROM table1 --sinistra
LEFT JOIN table2 --destra
ON table1.column_name = table2.column_name;

SELECT Orders.OrderID,
Customers.CustomerName
FROM Customers
LEFT JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
RIGHT (OUTER) JOIN Customers.CustomerName;
```

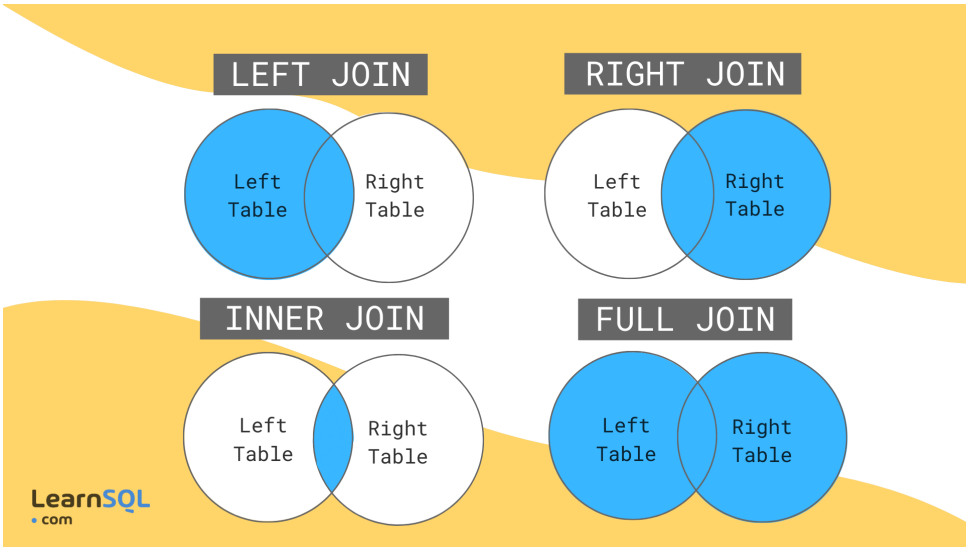
- FULL (OUTER) JOIN: unione completa

```
SELECT column_name(s)
FROM table1
CROSS JOIN table2; --unisce tutto

SELECT column_name(s) FROM table1
UNION --unisce sopra e sotto
SELECT column_name(s) FROM table2;

SELECT column_name(s) FROM table1
UNION ALL --come union ma include anche i duplicati
SELECT column_name(s) FROM table2;

SELECT City FROM Customers
UNION ALL
SELECT City FROM Suppliers
ORDER BY City;
```



ESERCIZIO 7

Si vogliono recuperare dal database "world" la lingua e la nazione di ogni città

```
SELECT city.Name, country.Name,
         countrylanguage.Language
FROM world.city
INNER JOIN world.country
ON city.CountryCode = country.Code
 INNER JOIN countrylanguage
 ON countrylanguage.CountryCode =  city.CountryCode;

SELECT city.Name AS CityName,
       country.Name AS CountryName,
       countrylanguage.Language
FROM world.city
INNER JOIN world.country
  ON city.CountryCode = country.Code
INNER JOIN world.countrylanguage
  ON countrylanguage.CountryCode = country.Code;
```

ESERCIZIO 8

Scrivete una query per recuperare il numero di città per nazione, numero ordinato in ordine decrescente

```
SELECT country.Name as CountryName,
        COUNT(city.Name) as CityNUmber
FROM country
JOIN city
ON city.CountryCode = country.Code
GROUP BY CountryName
ORDER BY CityNumber DESC
```

ESERCIZIO 9

Conoscere la lista delle repubbliche con aspettativa di vita maggiore di 70 anni, per ogni repubblica devono essere anche riportate le lingue parlate ufficiali

```
SELECT country.Name as CountryName,
        countrylanguage.Language as Country_Languages
FROM country
JOIN countrylanguage
ON countrylanguage.CountryCode = country.Code
WHERE
country.GovernmentForm LIKE '%Republic%'
 AND country.LifeExpectancy > 70
 AND countrylanguage.IsOfficial = 'T'
ORDER BY CountryName  ASC;
```

HAVING

Insieme a GROUP BY, filtra gruppi di righe

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

è come se fosse un WHERE sugli elementi raggruppati

EXISTS

verifica l'esistenza di records

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);

SELECT SupplierName
FROM Suppliers
WHERE EXISTS
(SELECT ProductName FROM Products WHERE Products.SupplierID =
Suppliers.supplierID AND Price < 20);
```

ANY

restituisce True o False se si soddisfano condizioni

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ANY
(SELECT column_name FROM table_name WHERE condition);

SELECT ProductName
FROM Products
WHERE ProductID = ANY
(SELECT ProductID FROM OrderDetails WHERE Quantity > 90);
```

ALL

restituisce True se tutti i record della tabella soddisfano una condizione

```
SELECT ALL column_name(s)
FROM table_name
WHERE column_name operator ALL
(SELECT column_name FROM table_name WHERE condition);

SELECT ProductName
FROM Products
WHERE ProductID = ALL (SELECT ProductID FROM OrderDetails WHE
```

ESERCIZIO 10

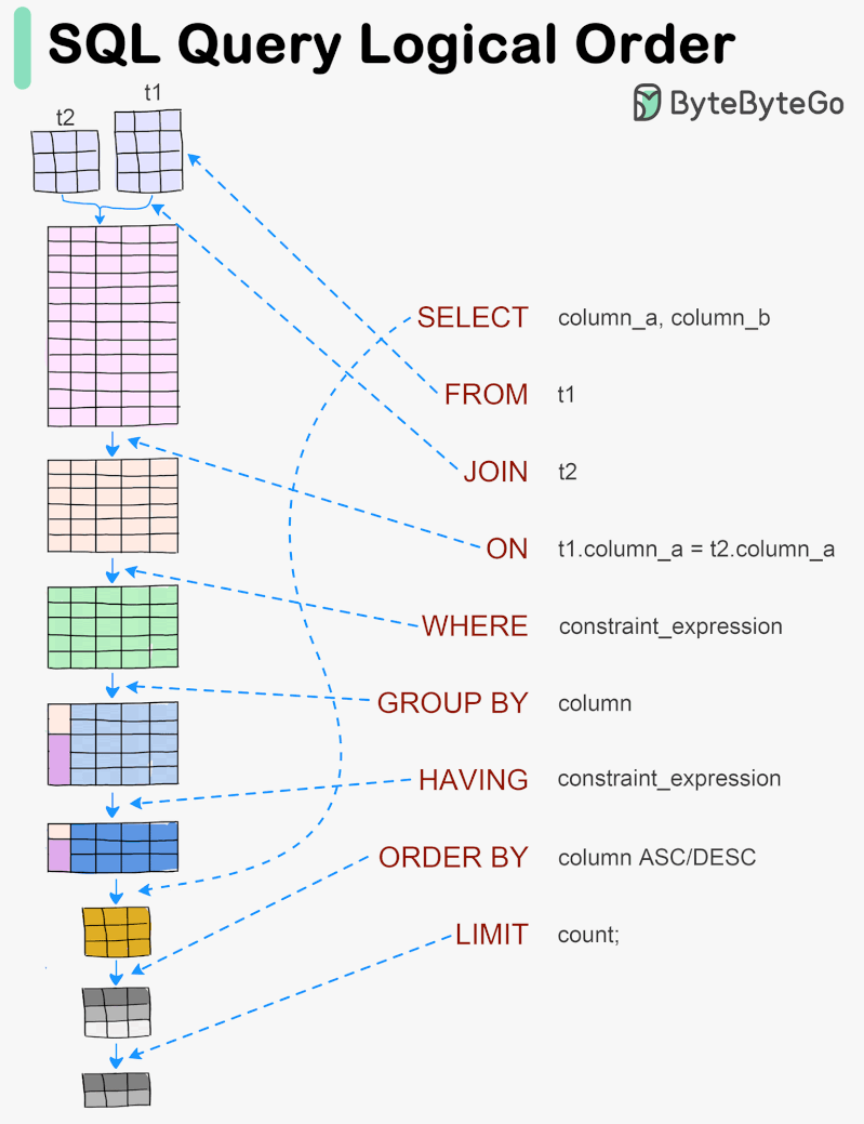
vogliamo recuperare dal database world le lingue più parlate per nazione con la rispettiva percentuale

```
SELECT country.Name as CountryName,
        countrylanguage.Language as Country_Languages,
        countrylanguage.Percentage
FROM country
JOIN countrylanguage
ON countrylanguage.CountryCode = country.Code
WHERE
Percentage >= 40
ORDER BY CountryName  ASC;
```

ESERCIZIO 11

Create una query che visualizzi il numero di popolazione per continente

```
SELECT country.Continent,
    SUM(country.Population) as Continent_Population
FROM country
GROUP BY Continent
ORDER BY Continent_Population DESC;
```



VIEW

Tabelle virtuali, è come se eseguisse in continuazione una query, crea tabelle dinamiche

```
-- Creazione di una vista che mostra
-- il nome e la popolazione dei paesi
CREATE VIEW CountryView AS
SELECT Name AS CountryName, Population
FROM country;

CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
```

```
WHERE condition;
DROP VIEW view_name;
```

INSERT INTO SELECT

```
INSERT INTO table2 (column1, column2, column3, ...)
SELECT column1, column2, column3, ...
FROM table1
WHERE condition;
```

CASE

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    WHEN conditionN THEN resultN
    ELSE result
END;

SELECT OrderID, Quantity,
CASE
    WHEN Quantity > 30 THEN 'The quantity is greater than 30'
    WHEN Quantity = 30 THEN 'The quantity is 30'
    ELSE 'The quantity is under 30'
END AS QuantityText
FROM OrderDetails;
```

```
SELECT CustomerName, City, Country
FROM Customers
ORDER BY
(CASE
    WHEN City IS NULL THEN Country
    ELSE City
END);
```

IFNULL()-COALESCE()

```
IFNULL(expression, alt_value)-- IFNULL()
-- è una funzione che restituisce
-- un valore alternativo sel'espressione è NULL:
COALESCE(val1, val2, ...., val_n) -- COALESCE()
-- è una funzione che restituisce
-- il primo valore non NULL:
```

ESERCIZIO 12

Create una vista chiamata CapitalCities che mostri il nome del paese e il nome della sua capitale di quel paese

```
CREATE VIEW CapitalCities AS
SELECT country.Name as Country,
        city.Name as CapitalCity
FROM country
JOIN city
ON city.ID = country.Capital
ORDER BY Country ASC
```