

## Il linguaggio Sql

Il linguaggio SQL è considerato il linguaggio standard per la gestione dei database relazionali. La sigla SQL sta per **Structured Query Language**, cioè linguaggio di richiesta strutturato.

Le sue istruzioni sono semplici ed intuitive e si suddividono in 2 tipologie:

- **DDL ( Data Definition Language )** *Linguaggio di definizione dei dati*, istruzioni per la definizione della struttura del database, delle relazioni e degli accessi al database
- **DML (Data Manipulation Language )** *Linguaggio di manipolazione dei dati*, istruzioni per modificare i dati i dati contenuti nel data base, con le operazioni di inserimento, modifica e cancellazione.

Le istruzioni Sql, vengono utilizzate in qualsiasi programma client Sql, tra cui Oracle, Informix, MySQL, SQLServer e ovviamente Access.

Questo infatti permette di creare il database, la struttura e le tabelle, attraverso una interfaccia grafica e le interrogazioni (query), attraverso la modalità QBE. E' però possibile creare sin dal principio il database utilizzando il linguaggio SQL, e più precisamente utilizzando le istruzioni della tipologia DDL.

### COMANDI DDL (Data Definition Language )

#### CREAZIONE TABELLE

```
CREATE TABLE nome_tabella
(campo1   tipo,
campo2   tipo,
campo3   tipo);
```

```
CREATE TABLE Studenti
(Cod_Stud smallint primary key,
Cognome  char(20),
Nome     char(15),
Città    char(20),
Età      smallint);
```

Sintassi

Esempio

L'esempio indica che la tabella Studenti, è costituita da 5 campi, Cod\_Stud è il campo che fungerà da chiave e conterrà valori interi, Cognome e Nome e Città contengono caratteri, rispettivamente di dimensione 20 e 15 ed età che conterrà valori interi.

Lo stesso esempio potrebbe essere arricchito di alcune clausole, ad esempio si potrebbe impostare che il campo Cognome e Nome non possano avere valore nullo, in fase di caricamento dei dati. Il codice diventerebbe:

```
CREATE TABLE Studenti
( Cod_Stud smallint primary Key,
Cognome  char(20) not null,
Nome     char(15) not null,
Città    char(20)
Età      smallint);
```

#### MODIFICA TABELLE

```
ALTER TABLE nome_tabella
ADD nome_attributo  —————> per aggiungere
                   oppure
```

```
ALTER TABLE Studenti
ADD Classe
                   aggiunge alla tabella Studenti il campo Classe
```

```
DROP nome_attributo  —————> per togliere
```

```
ALTER TABLE Studenti
DROP Classe
                   Toglie dalla tabella Studenti il campo Classe
```

**COMANDI DML (Data Manipulation Language)**

Per modificare i dati contenuti nel database, si utilizzano i comandi:

INSERT

↙

Inserire dati nel DB  
(Popolare un Database)

UPDATE


↓

Aggiorna dati del DB

DELETE

↘

Cancella dati del DB




**ESEMPIO : INSERIRE UNA RIGA NELLA TABELLA STUDENTI**

```
INSERT INTO Studenti
(Cod_Stud, Cognome, Nome, Città, Età)
VALUES ( 4, 'Rossi', 'Mario', 'Latina', 19);
```

```
INSERT INTO Studenti
(Cod_Stud, Cognome, Nome, Città, Età)
VALUES ( 5, 'Verdi', 'Giacomo', 'Roma', 18);
```

```
INSERT INTO Studenti
(Cod_Stud, Cognome, Nome, Città, Età)
VALUES ( 8, 'Bianchi', 'Angelo', 'Roma', 16);
```


| Cod_Stud | Cognome | Nome    | Città  | Età |
|----------|---------|---------|--------|-----|
| 4        | Rossi   | Mario   | Latina | 19  |
| 5        | Verdi   | Giacomo | Roma   | 18  |
| 8        | Bianchi | Angelo  | Roma   | 16  |
| *        |         |         |        |     |



**ESEMPIO : AGGIORNARE IL CAMPO CITTA' DELLO STUDENTE CON CODICE 4 DA "ROMA" A "LATINA"**

```
UPDATE Studenti
SET Città = "ROMA"
WHERE Cod_Stud =4;
```

| Cod_Stud | Cognome | Nome    | Città |
|----------|---------|---------|-------|
| 4        | Rossi   | Mario   | ROMA  |
| 5        | Verdi   | Giacomo | Roma  |
| 8        | Bianchi | Angelo  | Roma  |
| *        |         |         |       |



**ESEMPIO : ELIMINARE UNA RIGA DALLA TABELLA STUDENTI**

```
DELETE FROM Studenti
WHERE Cod_Stud =4;
```


| Cod_Stud | Cognome | Nome    | Città | Età |
|----------|---------|---------|-------|-----|
| 5        | Verdi   | Giacomo | Roma  | 18  |
| 8        | Bianchi | Angelo  | Roma  | 16  |
| *        |         |         |       |     |

**ESTRARRE DATI DA UN DATABASE CON IL LINGUAGGIO SQL: Le interrogazioni o Query**

Come con Access è possibile interrogare il database con le Query, anche con SQL si possono effettuare interrogazioni per estrarre dati dal database secondo criteri definiti.

```
SELECT Colonne
FROM Tabelle
WHERE Condizioni
```


L'istruzione SELECT, determina i campi che verranno estratti dalle tabelle indicate (FROM Tabelle) se soddisferanno i criteri indicati dopo il WHERE.



**ESEMPIO :ELENCO DEGLI STUDENTI RESIDENTI A LATINA**

```
SELECT Cognome, Nome
FROM Studenti
WHERE Città="LATINA";
```

| Cognome | Nome    |
|---------|---------|
| Gialli  | Romina  |
| Neri    | Rosanna |
| *       |         |



**ESEMPIO :ELENCO DEGLI STUDENTI MAGGIORENNI RESIDENTI A ROMA**

```
SELECT Cognome, Nome
FROM Studenti
WHERE Città="ROMA" and Et  >= 18;
```


| Cod_Studenti | Cognome | Nome    | Citt    | Et  |
|--------------|---------|---------|---------|-----|
| 5            | Verdi   | Giacomo | Roma    | 18  |
| 8            | Bianchi | Angelo  | Roma    | 16  |
| 10           | Gialli  | Romina  | Latina  | 17  |
| 12           | Rossi   | Mimmo   | Viterbo | 15  |
| 15           | Neri    | Rosanna | Latina  | 18  |
| *            |         |         |         |     |

Se **SELECT**   seguito dal simbolo \*, significa che si vogliono elencare tutti gli attributi del record selezionato. Quindi nell'esempio precedente, si potrebbe indicare

```
SELECT Studenti.*
FROM Studenti
WHERE Citt ="ROMA" and Et  >=18;
```

| Cod_Studenti | Cognome | Nome    | Citt  | Et  |
|--------------|---------|---------|-------|-----|
| 5            | Verdi   | Giacomo | Roma  | 18  |
| *            |         |         |       |     |

Il comando **SELECT** ha due predicati, **ALL** e **DISTINCT**.  
**ALL** è il predicato che l'istruzione Select ha di **default**, ed indica che la selezione prevede l'estrazione di **tutte le righe** che soddisfano il criterio della query, prevedendo dunque eventuali duplicazioni. Ad esempio se si volessero elencare le città di residenza degli Studenti, sicuramente, queste verrebbero ad essere duplicate nell'elenco qualora più studenti risiedessero nella stessa città. Per ovviare a tale duplicazione, si dovrà specificare il predicato **DISTINCT**, che eliminerà nell'elenco risultante dalla query, tutte le duplicazioni.

 **ESEMPIO :ELENCO DELLE CITTA' DEGLI STUDENTI.**

SELECT Città  
FROM Studenti

Il risultato sarà:

| Città   |
|---------|
| Roma    |
| Roma    |
| Latina  |
| Viterbo |
| Latina  |
| *       |

SELECT ALL Città  
FROM Studenti


Il risultato sarà:

| Città   |
|---------|
| Latina  |
| Roma    |
| Viterbo |

SELECT DISTINCT Città  
FROM Studenti

**APPROFONDIMENTI SULL'USO DI SELECT**

Ora vediamo alcuni esempi di interrogazioni in SQL, che rispecchino altri casi già elaborati con ACCESS. Ad esempio, valutiamo il codice della query che prevede oltre che l'estrazione di alcuni dati, la visualizzazione degli stessi cambiando il nome di un attributo.


 **ESEMPIO :ELENCO. DEGLI STUDENTI INDIVIDUATI DA Cod\_Stud, Nome e Cognome, VISUALIZZANDO LA MATRICOLA COME INTRESTAZIONE DELLA COLONNA CODICE.**

SELECT Cod\_Stud as Matricola, Nome, Cognome  
FROM Studenti;

| Matricola | Nome    | Cognome |
|-----------|---------|---------|
| 5         | Giacomo | Verdi   |
| 8         | Angelo  | Bianchi |
| 10        | Romina  | Gialli  |
| 12        | Mimmo   | Rossi   |
| 15        | Rosanna | Neri    |
| *         |         |         |


**QUERY PARAMETRICA IN SQL**

Di seguito un esempio di Query parametrizzata in codice:




**ESEMPIO :ELENCO DEGLI STUDENTI DI UNA CITTA' INDICATA DALL'UTENTE.**

```
SELECT Nome, Cognome
FROM Studenti
WHERE Città = [ Inserisci una città];
```



**OPERAZIONI RELAZIONALI IN SQL**

**CONGIUNZIONE**  
Il comando SELECT può operare su più tabelle, indicandone i nomi separati dalla virgola dopo la parola FROM e indicando dopo WHERE la condizione che dovranno soddisfare contemporaneamente sia le righe di una tabella che le righe di un'altra.



**ESEMPIO :ELENCO DEGLI STUDENTI CHE HANNO EFFETTUATO VERIFICHE CON RELATIVI DATI RIGUARDANTI LE VERIFICHE**

```
SELECT *
FROM Studenti, Verifiche
WHERE Cod_Stud= Codice_Studente;
```

```
SELECT *
FROM Studenti, Verifiche
WHERE Studenti.Cod_Stud = Verifiche.Codice_Studente
```

| Cod_Stud | Cognome | Nome    | Città   | Età | Cod_Verific | Codice_Stuc | Disciplina | Voto |
|----------|---------|---------|---------|-----|-------------|-------------|------------|------|
| 5        | Verdi   | Giacomo | Roma    | 18  | 5           | 5           | Storia     | 6    |
| 8        | Bianchi | Angelo  | Roma    | 16  | 3           | 8           | Italiano   | 5    |
| 8        | Bianchi | Angelo  | Roma    | 16  | 4           | 8           | Matematica | 7    |
| 8        | Bianchi | Angelo  | Roma    | 16  | 6           | 8           | Scienze    | 7    |
| 12       | Rossi   | Mimmo   | Viterbo | 15  | 7           | 12          | Inglese    | 5    |
| 15       | Neri    | Rosanna | Latina  | 18  | 8           | 15          | Inglese    | 9    |

Che QBE di Access traduce così:


```
SELECT Studenti.*, Verifiche.*
FROM Studenti INNER JOIN VERIFICHE ON Studenti.Cod_Stud = VERIFICHE.Codice_Studente;
```

**FUNZIONI DI AGGREGAZIONE IN SQL**

Il comando **SELECT** può contenere funzioni predefinite che operano sui valori di una singola colonna, restituendo un solo valore come potrebbe essere il minimo o il massimo dei valori considerati.

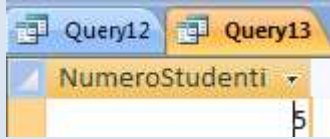
**Funzione COUNT**


La funzione **COUNT**, conta il numero delle righe selezionate



**ESEMPIO :CONTARE GLI STUDENTI**

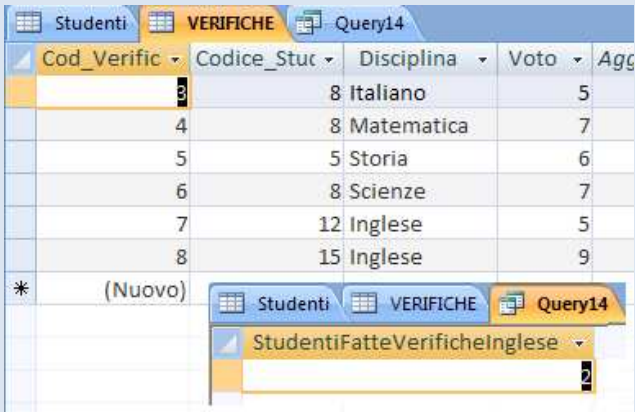
```
SELECT COUNT (*) AS NumeroStudenti
FROM Studenti;
```






**ESEMPIO : CONTARE GLI STUDENTI CHE HANNO EFFETTUATO VERIFICHE IN INGLESE**

```
SELECT COUNT (*) AS StudentiFatteVerificheInglese
FROM Verifiche
WHERE Disciplina = "inglese";
```



**Funzione SUM**  
La funzione **SUM**, produce la somma di tutti i valori (ovviamente numerici) contenuti nella colonna indicata.



**ESEMPIO :CALCOLARE LA SOMMA DEI VOTI CONTENUTI NELLA TABELLA VERIFICHE**

```
SELECT SUM (Voto)
FROM Verifiche
```

