INSERT

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);

INSERT INTO table_name (column1, column3, ...)
VALUES (value1, value3, ...);
```

L’istruzione INSERT **aggiunge una o più righe a una tabella.**

È anche possibile inserire dati solo in colonne specifiche.

UPDATE

```
UPDATE table_name SET column1 = value1, ...

UPDATE table_name SET column1 = value1, ...
WHERE condition;
```

L'istruzione **UPDATE** viene utilizzata per modificare i record esistenti in una tabella.

Nota: fare attenzione quando si aggiornano i record in una tabella! La clausola WHERE specifica quali record devono essere aggiornati. Se si omette la clausola WHERE, tutti i record nella tabella verranno aggiornati!

DELETE

```
DELETE FROM table_name;

DELETE FROM table_name WHERE condition;
```

L'istruzione **DELETE** viene utilizzata per eliminare record esistenti in una tabella.

Nota: fare attenzione quando si eliminano i record in una tabella! La clausola WHERE specifica quali record devono essere eliminati. Se si omette la clausola WHERE, tutti i record nella tabella verranno eliminati!

SELECT

L'istruzione *SELECT* viene utilizzata per **selezionare i dati da un database**.

Gli operatori ***UNION***, ***EXCEPT*** e ***INTERSECT*** possono essere utilizzati per combinare i risultati in un set di risultati.

```
SELECT *
FROM table_name;
```

```
SELECT column1, column2, ...
FROM table_name;
```

```
SELECT DISTINCT column1, ...
FROM table_name;
```

Demo

Inserire, aggiornare e eliminare dati

Select

icubed

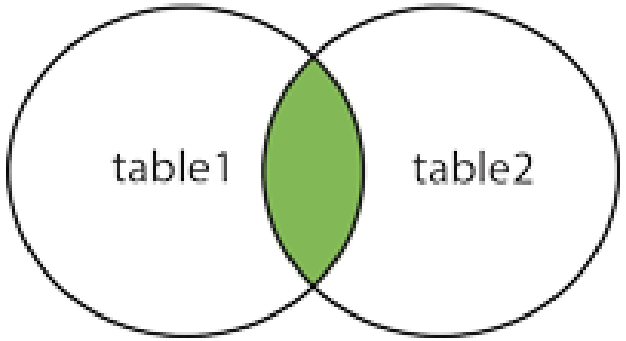
JOIN

Una clausola **JOIN** viene utilizzata per combinare righe da due o più tabelle, in base a una colonna correlata tra loro.

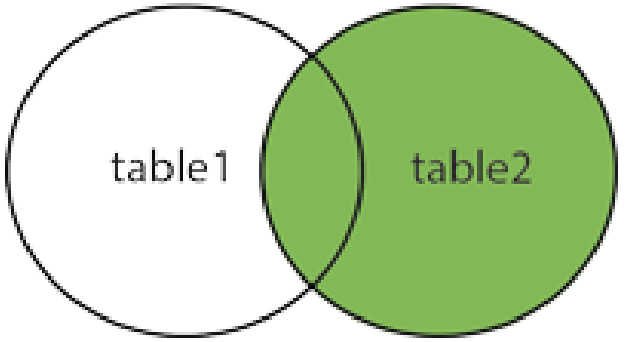
```
SELECT column1, column2, ...
FROM table1
INNER [ LEFT / RIGHT / FULL OUTER ] JOIN table2
ON table1.column_name = table2.column_name;
```

JOIN

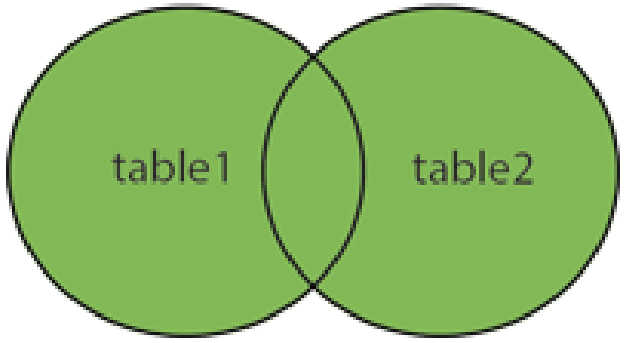
INNER JOIN



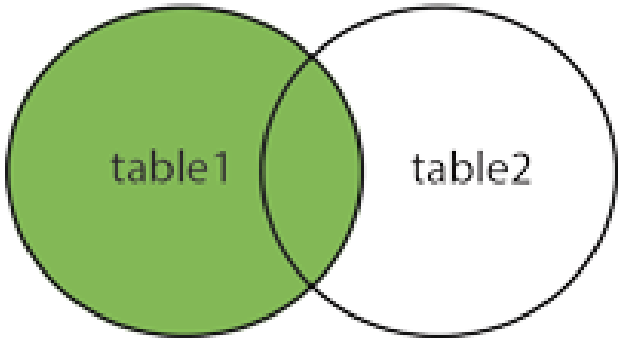
RIGHT JOIN



FULL OUTER JOIN



LEFT JOIN



icubed

ORDER BY

La parola chiave ORDER BY viene utilizzata per ordinare il set di risultati in ordine crescente o decrescente.

La parola chiave ORDER BY ordina i record in **ordine crescente** per impostazione predefinita.

Per ordinare i record in ordine decrescente, utilizzare la parola chiave DESC.

```
SELECT column1, column2, ...
FROM table1
ORDER BY column1, column2 DESC;
```

GROUP BY

L'istruzione GROUP BY **raggruppa righe con gli stessi valori in righe di riepilogo,** ad esempio "trova il numero di clienti in ciascun paese".

```
SELECT column1, ...
  FROM table_name
GROUP BY column_name
```

L'istruzione GROUP BY viene spesso utilizzata con funzioni aggregate (COUNT, MAX, MIN, SUM, AVG) per raggruppare il set di risultati per una o più colonne.

GROUP BY

- **MIN()** restituisce il valore più piccolo della colonna selezionata
- **MAX()** restituisce il valore più grande della colonna selezionata
- **COUNT()** restituisce il numero di righe che corrisponde a un criterio specificato

```
SELECT MIN(column1)
  FROM table_name
GROUP BY column_name
```

```
SELECT MAX(column1)
  FROM table_name
GROUP BY column_name
```

```
SELECT COUNT(column1)
  FROM table_name
GROUP BY column_name
```

GROUP BY

- **AVG()** restituisce il valore medio di una colonna numerica

```
SELECT AVG(column1)
  FROM table_name
GROUP BY column_name
```

- **SUM()** restituisce la somma totale di una colonna numerica

```
SELECT SUM(column1)
  FROM table_name
GROUP BY column_name
```

HAVING

La clausola **HAVING** è stata aggiunta a SQL perché non è possibile utilizzare la parola chiave WHERE con le funzioni di aggregazione.

```
SELECT column1, MAX(column2), ...
    FROM table_name
    GROUP BY column1
    HAVING COUNT(column2) > 0
```

IN

L'operatore **IN** consente di specificare più valori in una clausola **WHERE**.

È una scorciatoia per sostituire più condizioni OR.

```
SELECT column1, column2, ...
FROM table1
WHERE column_name = value1 OR column_name = value2 OR ...;
```

```
SELECT column1, column2, ...
FROM table1
WHERE column_name IN (value1, value2, ...);
```

BETWEEN

L'operatore **BETWEEN** seleziona i valori all'interno di un determinato intervallo. I valori possono essere numeri, testo o date.

È un operatore inclusivo, ovvero i *valori di inizio* e *fine* sono ***inclusi*** *nell'intervallo*.

```
SELECT column1, column2, ...
FROM table1
WHERE column_name BETWEEN value1 AND value2;
```

Demo

Join
Order By
Group by / Having
In
Between

icubed