CREAZIONE TABELLE (DDL - Data Definition Language)

CREATE TABLE

```
CREATE TABLE NomeTabella (
    campo1 TIPO [vincoli],
    campo2 TIPO [vincoli],
    ...
    PRIMARY KEY (campo_chiave),
    FOREIGN KEY (campo_esterno) REFERENCES AltraTabella(campo)
);
```

Esempio:

```
CREATE TABLE Studente (
    CodFiscale VARCHAR(16) PRIMARY KEY,
    Nome VARCHAR(50) NOT NULL,
    Cognome VARCHAR(50) NOT NULL,
    DataNascita DATE,
    Citta VARCHAR(50)
);
```

TIPI DI DATO PRINCIPALI

- `VARCHAR(n)` - Testo variabile
- `INT` - Numero intero
- `DATE` - Data (formato AAAA-MM-DD)
- `DECIMAL(n,d)` - Numero decimale

VINCOLI PRINCIPALI

- `PRIMARY KEY` - Chiave primaria
- `FOREIGN KEY` - Chiave esterna
- `NOT NULL` - Campo obbligatorio
- `UNIQUE` - Valore unico
- `AUTO_INCREMENT` - Valore automatico crescente

INTERROGAZIONI BASE (DQL - Data Query Language)

SELECT semplice

```sql
SELECT campo1, campo2, ...
FROM NomeTabella;

-- Selezionare tutti i campi
SELECT * FROM NomeTabella;

-- Dare un nome alternativo (ALIAS)
SELECT Nome AS NomeStudente, Cognome AS CognomeStudente
FROM Studente;
```

WHERE - Filtri sui dati

```sql
SELECT Nome, Cognome
FROM Studente
WHERE Citta = 'Roma';

-- Operatori di confronto: =, <>, !=, <, >, <=, >=
-- Operatori logici: AND, OR, NOT
```

Esempi pratici:

```sql
-- Studenti di Roma con età maggiore di 18
SELECT Nome, Cognome
FROM Studente
WHERE Citta = 'Roma' AND Eta > 18;

-- Studenti di Roma o Milano
SELECT Nome, Cognome
FROM Studente
WHERE Citta = 'Roma' OR Citta = 'Milano';

-- Età compresa tra 18 e 25 anni
SELECT Nome, Cognome
FROM Studente
WHERE Eta BETWEEN 18 AND 25;

-- Nomi che iniziano per 'M'
SELECT Nome, Cognome
FROM Studente
WHERE Nome LIKE 'M%';
```

FUNZIONI DI AGGREGAZIONE

Funzioni principali

```
COUNT(*)          -- Conta il numero di righe
COUNT(campo)      -- Conta i valori non nulli
SUM(campo)        -- Somma dei valori
AVG(campo)        -- Media dei valori
MAX(campo)        -- Valore massimo
MIN(campo)        -- Valore minimo
```

Esempi:

```
-- Numero totale di studenti
SELECT COUNT(*) AS NumeroStudenti
FROM Studente;

-- Età media degli studenti
SELECT AVG(Eta) AS EtaMedia
FROM Studente;

-- Età massima degli studenti di Roma
SELECT MAX(Eta) AS EtaMax
FROM Studente
WHERE Citta = 'Roma';
```

RAGGRUPPAMENTI

GROUP BY

```
SELECT campo_gruppo, FUNZIONE_AGGREGAZIONE(campo)
FROM NomeTabella
GROUP BY campo_gruppo;
```

Esempi:

```
-- Numero di studenti per città
SELECT Citta, COUNT(*) AS NumeroStudenti
FROM Studente
GROUP BY Citta;

-- Età media per città
```

```sql
SELECT Citta, AVG(Eta) AS EtaMedia
FROM Studente
GROUP BY Citta;
```

HAVING - Filtri sui gruppi

```sql
SELECT Citta, COUNT(*) AS NumeroStudenti
FROM Studente
GROUP BY Citta
HAVING COUNT(*) > 5;
```

Differenza WHERE vs HAVING:

- `WHERE` filtra le righe PRIMA del raggruppamento
- `HAVING` filtra i gruppi DOPO il raggruppamento

ORDINAMENTO

ORDER BY

```sql
SELECT Nome, Cognome, Eta
FROM Studente
ORDER BY Cognome ASC, Nome ASC;

-- ASC = crescente (default)
-- DESC = decrescente
```

SOTTOQUERY (SUBQUERY)

Query annidate

```sql
-- Studenti con età maggiore della media
SELECT Nome, Cognome
FROM Studente
WHERE Eta > (SELECT AVG(Eta) FROM Studente);

-- Studenti con età massima
SELECT Nome, Cognome
FROM Studente
WHERE Eta = (SELECT MAX(Eta) FROM Studente);
```

JOIN - UNIONI TRA TABELLE

INNER JOIN

```sql
SELECT s.Nome, s.Cognome, c.NomeCorso
FROM Studente s
INNER JOIN Iscrizione i ON s.CodFiscale = i.CodStudente
INNER JOIN Corso c ON i.CodCorso = c.CodCorso;
```

LEFT JOIN

```sql
-- Tutti gli studenti, anche quelli senza iscrizioni
SELECT s.Nome, s.Cognome, c.NomeCorso
FROM Studente s
LEFT JOIN Iscrizione i ON s.CodFiscale = i.CodStudente
LEFT JOIN Corso c ON i.CodCorso = c.CodCorso;
```

INSERIMENTO E MODIFICA DATI (DML)

INSERT

```sql
INSERT INTO Studente (CodFiscale, Nome, Cognome, DataNascita)
VALUES ('RSSMRA90A01H501X', 'Mario', 'Rossi', '1990-01-01');
```

UPDATE

```sql
UPDATE Studente
SET Citta = 'Milano'
WHERE CodFiscale = 'RSSMRA90A01H501X';
```

DELETE

```sql
DELETE FROM Studente
WHERE CodFiscale = 'RSSMRA90A01H501X';
```

PATTERN TIPICI PER L'ESAME

1. Elenco ordinato con filtri

```sql
SELECT Nome, Cognome, Voto
FROM Studente s
INNER JOIN Esame e ON s.CodFiscale = e.CodStudente
WHERE Materia = 'Informatica'
ORDER BY Voto DESC;
```

2. Statistiche per gruppi

```sql
SELECT Citta, COUNT(*) AS NumStudenti, AVG(Voto) AS VotoMedio
FROM Studente s
INNER JOIN Esame e ON s.CodFiscale = e.CodStudente
GROUP BY Citta
HAVING AVG(Voto) > 7;
```

3. Studenti che NON hanno mai fatto qualcosa

```sql
SELECT Nome, Cognome
FROM Studente
WHERE CodFiscale NOT IN (
    SELECT CodStudente FROM Esame
);
```

4. Conteggi con date

```sql
SELECT COUNT(*) AS EsamiAnnoCorrente
FROM Esame
WHERE YEAR(DataEsame) = YEAR(NOW());
```

TRUCCHI PER L'ESAME

1. **Sempre usare alias per le tabelle:** `FROM Studente s`
2. **Controllare sempre i JOIN:** verificare le chiavi di collegamento
3. **WHERE prima di GROUP BY:** i filtri sui dati vanno prima del raggruppamento
4. **HAVING per i gruppi:** i filtri sui risultati aggregati vanno dopo GROUP BY
5. **Controllare i tipi di dato:** DATE va in formato 'AAAA-MM-DD'
6. **DISTINCT per eliminare duplicati:** `SELECT DISTINCT campo`

ESEMPI COMPLETI

Schema tipico d'esame

```sql
-- Tabelle di esempio
CREATE TABLE Studente (
    Id INT PRIMARY KEY AUTO_INCREMENT,
    Nome VARCHAR(50),
    Cognome VARCHAR(50),
    Eta INT,
    Citta VARCHAR(50)
);

CREATE TABLE Corso (
    Id INT PRIMARY KEY AUTO_INCREMENT,
    Nome VARCHAR(50),
    Crediti INT
);

CREATE TABLE Iscrizione (
    IdStudente INT,
    IdCorso INT,
    DataIscrizione DATE,
    Voto INT,
    FOREIGN KEY (IdStudente) REFERENCES Studente(Id),
    FOREIGN KEY (IdCorso) REFERENCES Corso(Id)
);
```

Query complessa tipica

```sql
-- Studenti con voto medio superiore a 25, ordinati per città
SELECT s.Nome, s.Cognome, s.Citta, AVG(i.Voto) AS VotoMedio
FROM Studente s
INNER JOIN Iscrizione i ON s.Id = i.IdStudente
GROUP BY s.Id, s.Nome, s.Cognome, s.Citta
HAVING AVG(i.Voto) > 25
ORDER BY s.Citta, s.Cognome;
```