Il Linguaggio SQL

Costrutti DDL - Data Definition Language

Prof. Massimo Fedeli

ITS Made in Italy - Fabbrica Digitale

9 gennaio 2026

Sommario

Data Definition Language (DDL)

Definizione

Il DDL è l'insieme di istruzioni utilizzate per **modificare la struttura** della base di dati.

Operazioni principali:

- Definizione di schemi
- Creazione di tabelle
- Modifica di strutture
- Cancellazione di oggetti

Elementi gestiti:

- Tabelle (table)
- Vincoli (constraints)
- Domini
- Viste (view)

DDL vs DML

sqlblue!20 **DDL (Definition)**	**DML (Manipulation)**
Modifica la **struttura**	Modifica i **dati**
CREATE, ALTER, DROP	INSERT, UPDATE, DELETE, SELECT
Definisce schemi e tabelle	Opera sulle righe
Effetto permanente	Può essere annullato (ROLLBACK)

Attenzione

Le operazioni DDL sono **irreversibili** e modificano la struttura del database in modo permanente.

Creazione di uno Schema

Sintassi

```
CREATE SCHEMA [IF NOT EXISTS] NomeSchema
[AUTHORIZATION utente];
```

Esempi pratici:

```
-- Creazione schema semplice
CREATE SCHEMA IF NOT EXISTS UniversitaDB;

-- Creazione con autorizzazione
CREATE SCHEMA AziendaDB AUTHORIZATION amministratore;
```

Note

- La clausola `IF NOT EXISTS` evita errori se lo schema esiste già
- Uno schema è una collezione logica di tabelle e oggetti

Selezione e Cancellazione Schema

Selezionare uno schema:

```
USE NomeSchema;
```

Cancellare uno schema:

```
DROP SCHEMA [IF EXISTS] NomeSchema;
```

Attenzione!

Il comando `DROP SCHEMA` cancella **tutto il contenuto** dello schema, incluse tutte le tabelle e i dati in esse contenuti.

Esempio completo:

```
USE UniversitaDB;  -- Seleziona lo schema
-- ... operazioni varie ...
DROP SCHEMA IF EXISTS TestDB;  -- Cancella schema di test
```

CREATE TABLE - Sintassi Base

Sintassi

```
CREATE TABLE [IF NOT EXISTS] NomeTabella (
nome_campo1 tipo_campo1 [vincoli],
nome_campo2 tipo_campo2 [vincoli],
...
[vincoli_tabella]
);
```

Esempio basilare:

```
CREATE TABLE Studenti (
Matricola CHAR(10),
Nome VARCHAR(20),
Cognome VARCHAR(20),
AnnoIscrizione INTEGER
);
```

Tipi di Dato in SQL

Tipi numerici:

- `INTEGER` / `INT`
- `SMALLINT`
- `DECIMAL(p,s)` / `NUMERIC`
- `FLOAT` / `REAL`
- `DOUBLE`

Tipi stringa:

- `CHAR(n)` - lunghezza fissa
- `VARCHAR(n)` - lunghezza variabile
- `TEXT` - testo lungo

Altri tipi:

- `DATE`
- `TIME`
- `DATETIME` / `TIMESTAMP`
- `BOOLEAN`

Esempio Completo di CREATE TABLE

```
CREATE TABLE IF NOT EXISTS Studenti (
Matricola CHAR(10),
Nome VARCHAR(20) NOT NULL,
Cognome VARCHAR(20) NOT NULL,
CodiceFiscale CHAR(16) UNIQUE,
DataNascita DATE,
AnnoIscrizione INTEGER DEFAULT 1,
Email VARCHAR(100),
CONSTRAINT pk_matricola PRIMARY KEY (Matricola)
);
```

Elementi chiave

- `NOT NULL`: impedisce valori nulli
- `UNIQUE`: valori univoci nella colonna
- `DEFAULT`: valore predefinito
- `PRIMARY KEY`: chiave primaria

Valori di Default

Definizione

I valori di default specificano cosa assegnare all'attributo quando non si indica un valore esplicitamente.

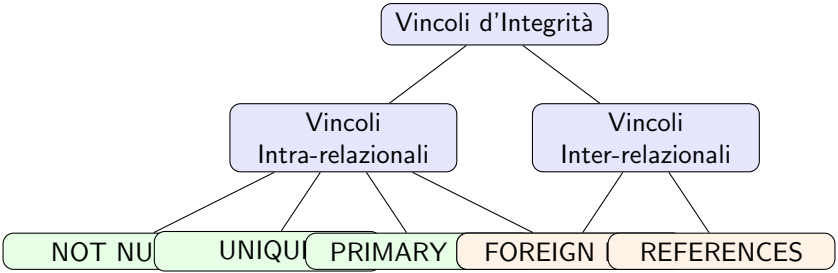
Esempi:

```
CREATE TABLE Iscrizioni (
ID INTEGER PRIMARY KEY,
DataIscrizione DATE DEFAULT CURRENT_DATE,
AnnoAccademico INTEGER DEFAULT 1,
NumeroPatente CHAR(20) DEFAULT NULL,
Stato VARCHAR(20) DEFAULT 'ATTIVO'
);
```

Nota

Se non si specifica un valore di default, si assume `NULL` (se ammesso).

Tipologie di Vincoli



Vincolo NOT NULL

Scopo

Vieta la presenza di valori nulli in quella colonna.

Sintassi:

```
CREATE TABLE Studenti (
Matricola CHAR(10),
Nome VARCHAR(20) NOT NULL DEFAULT 'N.D.',
Cognome VARCHAR(20) NOT NULL,
Email VARCHAR(100),  -- può essere NULL
PRIMARY KEY (Matricola)
);
```

Importante

- Il vincolo NOT NULL si applica a **singole colonne**
- Il valore di default viene specificato **dopo** il vincolo NOT NULL
- Non si può specificare NOT NULL per coppie di colonne

Vincolo UNIQUE

Scopo

Garantisce che non esistano due righe con gli stessi valori per l'attributo specificato.

UNIQUE su singola colonna:

```
CREATE TABLE Studenti (
Matricola CHAR(10),
CodiceFiscale CHAR(16) UNIQUE,
Email VARCHAR(100) NOT NULL UNIQUE,
Nome VARCHAR(20) NOT NULL,
Cognome VARCHAR(20) NOT NULL,
PRIMARY KEY (Matricola)
);
```

Nota su NULL

Il vincolo UNIQUE **non esclude** la presenza di più righe con valori NULL (che si assumono tutti diversi fra loro).

Vincolo UNIQUE su Più Colonne

Sintassi 1: Senza nome vincolo

```
CREATE TABLE Studenti (
Matricola CHAR(10),
Nome VARCHAR(20) NOT NULL,
Cognome VARCHAR(20) NOT NULL,
AnnoIscrizione INTEGER,
UNIQUE(Nome, Cognome),
PRIMARY KEY (Matricola)
);
```

Sintassi 2: Con nome vincolo

```
CREATE TABLE Studenti (
Matricola CHAR(10),
Nome VARCHAR(20) NOT NULL,
Cognome VARCHAR(20) NOT NULL,
AnnoIscrizione INTEGER,
CONSTRAINT uq_nome_cognome UNIQUE(Nome, Cognome),
PRIMARY KEY (Matricola)
);
```

Ordine dei Vincoli

Regola

Quando i vincoli sono espressi su una singola colonna, l'ordine è:

$$\texttt{tipo} \rightarrow \texttt{NOT NULL} \rightarrow \texttt{DEFAULT} \rightarrow \texttt{UNIQUE}$$

Esempio corretto:

```
CREATE TABLE Studenti (
Matricola CHAR(10),
Nome VARCHAR(20) NOT NULL DEFAULT 'N.D.' UNIQUE,
Cognome VARCHAR(20) NOT NULL,
AnnoIscrizione INTEGER,
PRIMARY KEY (Matricola)
);
```

Attenzione

Un ordine errato può causare errori di sintassi.

Vincolo PRIMARY KEY

Scopo

Identifica la chiave primaria della tabella (implica NOT NULL e UNIQUE).

Può esserci un solo vincolo di PRIMARY KEY per tabella!

Sintassi 1: Chiave primaria singola

```
CREATE TABLE Studenti (
Matricola CHAR(10) PRIMARY KEY,
Nome VARCHAR(20) NOT NULL,
Cognome VARCHAR(20) NOT NULL
);
```

Sintassi 2: Chiave primaria composta

```
CREATE TABLE Veicoli (
Targa CHAR(10),
CodiceProprietario CHAR(20) NOT NULL,
PRIMARY KEY (Targa, CodiceProprietario)
);
```

PRIMARY KEY - Tre Modalità

Modalità 1: Inline (solo chiave singola)

```
CREATE TABLE Veicoli (
Targa CHAR(10) PRIMARY KEY,
CodiceProprietario CHAR(20) NOT NULL
);
```

Modalità 2: A livello di tabella

```
CREATE TABLE Veicoli (
Targa CHAR(10),
CodiceProprietario CHAR(20) NOT NULL,
PRIMARY KEY (Targa, CodiceProprietario)
);
```

Modalità 3: Con nome vincolo

```
CREATE TABLE Veicoli (
Targa CHAR(10),
CodiceProprietario CHAR(20) NOT NULL,
CONSTRAINT pk_veicoli PRIMARY KEY (Targa, CodiceProprietario)
);
```

Vincolo CHECK

Scopo

Verifica generiche condizioni sui valori di una o più colonne.

CHECK su singola colonna:

```
CREATE TABLE Esami (
Matricola CHAR(10),
Corso VARCHAR(20) UNIQUE,
Voto INTEGER CHECK (Voto >= 18 AND Voto <= 30),
PRIMARY KEY (Corso, Matricola)
);
```

CHECK su più colonne:

```
CREATE TABLE Fatture (
ID INTEGER PRIMARY KEY,
ImportoLordo DECIMAL(10,2),
Netto DECIMAL(10,2),
Ritenute DECIMAL(10,2),
CHECK (ImportoLordo = Netto + Ritenute)
```

CHECK - Esempi Avanzati

Con nome vincolo:

```
CREATE TABLE Esami (
Matricola CHAR(10),
Corso VARCHAR(20),
Voto INTEGER,
Lode BOOLEAN DEFAULT FALSE,
CONSTRAINT ck_voto CHECK (Voto >= 18 AND Voto <= 30),
CONSTRAINT ck_lode CHECK (Lode = FALSE OR Voto = 30),
PRIMARY KEY (Corso, Matricola)
);
```

Nota

Il vincolo è violato se esiste **almeno una tupla** che rende falsa la condizione.

Altri esempi:

```
CHECK (DataFine > DataInizio)
CHECK (Stipendio > 0)
CHECK (Email LIKE '%@%.%')
```

Regola

L'attributo della tabella esterna deve essere soggetto a vincolo UNIQUE (di solito PRIMARY KEY).

Definizione

Vincolo che crea un legame tra valori di attributi in tabelle diverse.

FOREIGN KEY - Concetto

Vincolo di Foreign Key

Impone che, per ogni tupla, il valore dell'attributo della tabella interna, se diverso da NULL, deve essere uguale a un valore dell'attributo della tabella esterna.

Schema esempio:

```
-- Tabella esterna (riferita)
CREATE TABLE Dipartimenti (
NomeDipartimento CHAR(15) PRIMARY KEY,
Sede VARCHAR(50)
);

-- Tabella interna (referente)
CREATE TABLE Impiegati (
Matricola CHAR(6) PRIMARY KEY,
Nome VARCHAR(50) NOT NULL,
Cognome VARCHAR(50) NOT NULL,
Dipartimento CHAR(15),
FOREIGN KEY (Dipartimento)
```

Vincolo REFERENCES

Uso

Permette di specificare vincoli di colonna per singoli attributi.

Sintassi compatta:

```
CREATE TABLE Impiegati (
Matricola CHAR(6) PRIMARY KEY,
Cognome VARCHAR(50) NOT NULL,
Nome VARCHAR(50) NOT NULL,
Dipartimento CHAR(15)
REFERENCES Dipartimenti(NomeDipartimento),
Stipendio DECIMAL(10,2)
);
```

Quando usare REFERENCES

- Quando la foreign key è composta da **un solo attributo**
- Per una sintassi più compatta e leggibile

FOREIGN KEY per Più Attributi

Vincolo FOREIGN KEY

Necessario quando si referenziano più attributi contemporaneamente.

```
CREATE TABLE Anagrafica (
Nome VARCHAR(50),
Cognome VARCHAR(50),
DataNascita DATE,
PRIMARY KEY (Nome, Cognome)
);

CREATE TABLE Impiegati (
Matricola CHAR(6) PRIMARY KEY,
NomeDip VARCHAR(50),
CognomeDip VARCHAR(50),
Stipendio DECIMAL(10,2),
FOREIGN KEY (NomeDip, CognomeDip)
REFERENCES Anagrafica(Nome, Cognome)
);
```

FOREIGN KEY con Nome Vincolo

Best Practice

Dare un nome al vincolo facilita successive modifiche e debugging.

```
CREATE TABLE Impiegati (
Matricola CHAR(6) PRIMARY KEY,
NomeDip VARCHAR(50),
CognomeDip VARCHAR(50),
Dipartimento CHAR(15),
CONSTRAINT fk_anagrafica
FOREIGN KEY (NomeDip, CognomeDip)
REFERENCES Anagrafica(Nome, Cognome),
CONSTRAINT fk_dipartimento
FOREIGN KEY (Dipartimento)
REFERENCES Dipartimenti(NomeDipartimento)
);
```

Vantaggio

Il nome permette di rimuovere facilmente il vincolo con ALTER TABLE ... DROP CONSTRAINT

Politiche di Reazione - Introduzione

Problema

Cosa succede quando si modifica o cancella una riga nella tabella esterna che è referenziata da una foreign key?

Tabella interna:

- Inserimento: sempre controllato
- Modifica: sempre controllata
- Nessuna politica necessaria

Tabella esterna:

- Modifica: politica definibile
- Cancellazione: politica definibile
- 4 opzioni disponibili

Default

Se non specificata, la politica predefinita è `NO ACTION` (o `RESTRICT`).

Sintassi Politiche di Reazione

Sintassi

```
FOREIGN KEY (colonna) REFERENCES tabella(colonna)
ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }
ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }
```

Le quattro politiche:

1. `NO ACTION` (o `RESTRICT`): blocca l'operazione
2. `CASCADE`: propaga l'operazione
3. `SET NULL`: imposta NULL
4. `SET DEFAULT`: imposta valore di default

MySQL

La politica `SET DEFAULT` **non è supportata** in MySQL.

Politiche ON UPDATE

NO ACTION / RESTRICT

Se si tenta di aggiornare un valore referenziato, l'operazione viene **bloccata** e viene generato un errore.

CASCADE

Se si aggiorna un valore nella tabella esterna, tutti i valori corrispondenti nella tabella interna vengono **aggiornati automaticamente** al nuovo valore.

SET NULL

Se si aggiorna un valore nella tabella esterna, i valori corrispondenti nella tabella interna vengono impostati a `NULL`.

SET DEFAULT

I valori corrispondenti nella tabella interna vengono impostati al **valore di default** (non supportato in MySQL).

Politiche ON DELETE

NO ACTION / RESTRICT

Se si tenta di cancellare una riga referenziata, l'operazione viene **bloccata** e viene generato un errore.

CASCADE

Se si cancella una riga nella tabella esterna, tutte le righe corrispondenti nella tabella interna vengono **cancellate automaticamente**.

SET NULL

Se si cancella una riga nella tabella esterna, i valori della foreign key nella tabella interna vengono impostati a `NULL`.

SET DEFAULT

I valori della foreign key nella tabella interna vengono impostati al **valore di default** (non supportato in MySQL).

Esempio CASCADE

```sql
CREATE TABLE Dipartimenti (
IdReparto INTEGER PRIMARY KEY,
Nome VARCHAR(50)
);

CREATE TABLE Impiegati (
Matricola CHAR(6) PRIMARY KEY,
Nome VARCHAR(50),
IdReparto INTEGER,
CONSTRAINT fk_reparto FOREIGN KEY (IdReparto)
REFERENCES Dipartimenti(IdReparto)
ON DELETE CASCADE
ON UPDATE CASCADE
);
```

Comportamento

- Se si cancella un dipartimento, vengono cancellati **tutti gli impiegati** di quel dipartimento
- Se si modifica l'ID di un dipartimento, viene aggiornato automaticamente in tutti gli impiegati

Esempio SET NULL

```
CREATE TABLE Impiegati (
Matricola CHAR(6) PRIMARY KEY,
Nome VARCHAR(50),
IdReparto INTEGER,
CONSTRAINT fk_reparto FOREIGN KEY (IdReparto)
REFERENCES Dipartimenti(IdReparto)
ON DELETE SET NULL
ON UPDATE NO ACTION
);
```

Comportamento

- Se si cancella un dipartimento, il campo `IdReparto` degli impiegati viene impostato a `NULL`
- Le modifiche all'ID del dipartimento sono **bloccate** se ci sono impiegati collegati

Requisito

La colonna `IdReparto` deve **permettere NULL**.

Esempio Completo con Politiche

```sql
CREATE TABLE Studenti (
Matricola CHAR(10) PRIMARY KEY,
Nome VARCHAR(50) NOT NULL,
Cognome VARCHAR(50) NOT NULL
);

CREATE TABLE Iscrizioni (
CodIscrizione CHAR(20),
Matricola CHAR(10),
CorsoID INTEGER,
CONSTRAINT pk_iscrizioni
PRIMARY KEY (CodIscrizione, Matricola),
CONSTRAINT fk_studenti
FOREIGN KEY (Matricola)
REFERENCES Studenti(Matricola)
ON DELETE CASCADE        -- cancellazione in cascata
ON UPDATE NO ACTION      -- modifiche non permesse
);
```

DROP TABLE

Scopo

Rimuove completamente una tabella dal database.

Sintassi:

```
DROP TABLE [IF EXISTS] NomeTabella [RESTRICT | CASCADE];
```

RESTRICT:

- Cancella solo se vuota
- Nessun oggetto dipendente
- (MySQL: non implementato)

CASCADE:

- Cancella tabella e dati
- Cancella viste dipendenti
- (MySQL: non implementato)

MySQL

In MySQL, `DROP TABLE` cancella sempre la tabella e tutti i suoi dati, ignorando RESTRICT/CASCADE.

DROP TABLE - Esempi

Esempi pratici:

```sql
-- Cancellazione semplice
DROP TABLE Studenti;

-- Cancellazione sicura (no errore se non esiste)
DROP TABLE IF EXISTS StudentiTemp;

-- Cancellazione di pi  tabelle
DROP TABLE IF EXISTS Tabella1, Tabella2, Tabella3;
```

Attenzione alle Foreign Key

Se altre tabelle hanno foreign key che riferiscono la tabella da cancellare:

- L'operazione viene bloccata
- Soluzione: disabilitare temporaneamente i controlli

```sql
SET FOREIGN_KEY_CHECKS = 0;
DROP TABLE TabellaEsterna;
```

DROP TABLE vs DELETE FROM

sqlblue!20 **DROP TABLE Tabella**	**DELETE FROM Tabella**
Elimina la **struttura**	Elimina solo i **dati**
Cancella lo schema	Mantiene lo schema
Rimuove vincoli e indici	Mantiene vincoli e indici
Cancella viste dipendenti	Mantiene le viste
Irreversibile	Può essere annullato (ROLLBACK)
Più veloce	Più lento

Quando usare cosa?

- `DROP TABLE`: quando non serve più la tabella
- `DELETE FROM`: quando si vogliono cancellare solo i dati

ALTER TABLE - Introduzione

Scopo

Permette di modificare la struttura di una tabella esistente.

Operazioni possibili:

- Aggiungere colonne
- Rimuovere colonne
- Modificare colonne
- Aggiungere vincoli
- Rimuovere vincoli

Sintassi base:

```
ALTER TABLE NomeTabella
ADD [COLUMN] NomeColonna Definizione;

ALTER TABLE NomeTabella
DROP [COLUMN] NomeColonna;
```

ALTER TABLE - Aggiungere Colonne

Aggiunta colonna semplice:

```
ALTER TABLE Studenti
ADD COLUMN Sesso CHAR(1);
```

Aggiunta con vincoli e default:

```
ALTER TABLE Studenti
ADD COLUMN DataIscrizione DATE DEFAULT CURRENT_DATE;

ALTER TABLE Studenti
ADD COLUMN Email VARCHAR(100) UNIQUE;

ALTER TABLE Studenti
ADD COLUMN Telefono VARCHAR(15) NOT NULL DEFAULT 'N.D.';
```

Nota

Le nuove colonne assumono valore `NULL` o il valore di default per tutte le righe esistenti.

ALTER TABLE - Rimuovere Colonne

Rimozione colonna:

```
ALTER TABLE Studenti
DROP COLUMN AnnoIscrizione;
```

Attenzione!

- La cancellazione è **irreversibile**
- Se la colonna è referenziata da foreign key, l'operazione **fallisce**
- Tutti i dati nella colonna vengono **persi**

Esempio che fallisce:

```
CREATE TABLE BaseTabella (
ID INTEGER PRIMARY KEY,
Nome VARCHAR(50) UNIQUE
);

-- Fallisce se Nome    referenziato
ALTER TABLE BaseTabella DROP COLUMN Nome;
```

ALTER TABLE - Gestione Vincoli (1)

Aggiungere vincolo di PRIMARY KEY:

```
ALTER TABLE Studenti
ADD PRIMARY KEY (Matricola);
```

Rimuovere PRIMARY KEY:

```
ALTER TABLE Studenti
DROP PRIMARY KEY;
```

Aggiungere UNIQUE:

```
ALTER TABLE Studenti
ADD CONSTRAINT uq_email UNIQUE (Email);
```

Rimuovere UNIQUE:

```
ALTER TABLE Studenti
DROP INDEX uq_email;
-- oppure
ALTER TABLE Studenti
DROP KEY uq_email;
```

ALTER TABLE - Gestione Vincoli (2)

Aggiungere FOREIGN KEY:

```
ALTER TABLE Impiegati
ADD CONSTRAINT fk_dipartimento
FOREIGN KEY (IdReparto)
REFERENCES Dipartimenti(IdReparto)
ON DELETE CASCADE;
```

Rimuovere FOREIGN KEY:

```
ALTER TABLE Impiegati
DROP FOREIGN KEY fk_dipartimento;
```

Importante

- Quando si aggiunge un vincolo, **deve essere soddisfatto** dall'istanza corrente
- Usare nomi espliciti per i vincoli facilita la loro rimozione

ALTER TABLE - Esempio Completo

```sql
-- Creazione tabella base
CREATE TABLE Studenti (
Matricola CHAR(10),
Nome VARCHAR(50)
);

-- Aggiunta PRIMARY KEY
ALTER TABLE Studenti ADD PRIMARY KEY (Matricola);

-- Aggiunta colonne
ALTER TABLE Studenti ADD COLUMN Cognome VARCHAR(50);
ALTER TABLE Studenti ADD COLUMN Email VARCHAR(100) UNIQUE;

-- Aggiunta vincolo NOT NULL (MySQL: modifica colonna)
ALTER TABLE Studenti MODIFY Cognome VARCHAR(50) NOT NULL;

-- Aggiunta FOREIGN KEY
ALTER TABLE Studenti
ADD COLUMN CorsoID INTEGER,
```

Viste - Introduzione

Definizione

Le viste sono **tabelle virtuali** il cui contenuto dipende dal contenuto di altre tabelle.

Sintassi:

```
CREATE [OR REPLACE] VIEW NomeVista [(ListaAttributi)]
AS SelectSQL;
```

Esempio base:

```
CREATE VIEW ImpiegatiCostosi (Matricola, Nome, Cognome)
AS SELECT Matricola, Nome, Cognome
FROM Impiegati
WHERE StipendioAnnuale > 30000;
```

Nota

La lista degli attributi può essere omessa. In tal caso, lo schema è quello della SELECT.

Utilizzo delle Viste

Una volta creata, la vista si usa come una normale tabella:

```sql
-- Interrogazione della vista
SELECT * FROM ImpiegatiCostosi;

-- Join con la vista
SELECT IC.Nome, IC.Cognome, D.Nome AS Dipartimento
FROM ImpiegatiCostosi IC
JOIN Dipartimenti D ON IC.IdReparto = D.IdReparto;

-- Filtri sulla vista
SELECT * FROM ImpiegatiCostosi
WHERE Nome LIKE 'M%'
ORDER BY Cognome;
```

Vantaggio

Le viste permettono di semplificare query complesse e riutilizzarle facilmente.

Viste - Esempi Pratici

Vista per impiegati recenti:

```
CREATE VIEW ImpiegatiRecenti (Matricola, Nome, Cognome, DataAssunzione
    )
AS SELECT Matricola, Nome, Cognome, DataAssunzione
FROM Impiegati
WHERE DataAssunzione > '2020-01-01';
```

Vista con join:

```
CREATE VIEW ImpiegatiMilanesi
AS SELECT I.Matricola, I.Nome, I.Cognome, I.StipendioAnnuale
FROM Impiegati I
JOIN Reparti R ON I.IdReparto = R.IdReparto
WHERE R.Citta = 'Milano';
```

Uso della vista:

```
-- Media stipendi impiegati milanesi
SELECT AVG(StipendioAnnuale) FROM ImpiegatiMilanesi;
```

Viste - Esempi con Aggregazione

Vista con GROUP BY:

```sql
CREATE VIEW StatisticheDipartimenti
AS SELECT
D.Nome AS Dipartimento,
COUNT(*) AS NumeroImpiegati,
AVG(I.StipendioAnnuale) AS StipendioMedio,
MAX(I.StipendioAnnuale) AS StipendioMax
FROM Dipartimenti D
JOIN Impiegati I ON D.IdReparto = I.IdReparto
GROUP BY D.IdReparto, D.Nome;
```

Interrogazione della vista:

```sql
-- Dipartimenti con pi  di 10 impiegati
SELECT * FROM StatisticheDipartimenti
WHERE NumeroImpiegati > 10;

-- Dipartimenti ordinati per stipendio medio
SELECT * FROM StatisticheDipartimenti
ORDER BY StipendioMedio DESC;
```

Utilizzi Tipici delle Viste

1. **Sicurezza:**
   - Impedire ad alcuni utenti l'accesso completo ai dati
   - Esempio: vista che nasconde stipendi e dati sensibili
2. **Convenienza:**
   - Semplificare la scrittura delle query
   - Quando una query complessa compare più volte
3. **Indipendenza logica:**
   - Isolare le applicazioni dalla struttura fisica del DB
   - Se cambia lo schema, le applicazioni continuano a funzionare
4. **Riutilizzo del codice:**
   - Definire una volta, usare ovunque
   - Manutenzione centralizzata della logica di business

Modifica e Cancellazione Viste

Modificare una vista esistente:

```sql
CREATE OR REPLACE VIEW ImpiegatiCostosi
AS SELECT Matricola, Nome, Cognome, StipendioAnnuale
FROM Impiegati
WHERE StipendioAnnuale > 40000;  -- soglia modificata
```

Cancellare una vista:

```sql
DROP VIEW [IF EXISTS] NomeVista;
```

Esempi:

```sql
DROP VIEW ImpiegatiCostosi;

DROP VIEW IF EXISTS ImpiegatiRecenti;

-- Cancellazione multipla
DROP VIEW IF EXISTS Vista1, Vista2, Vista3;
```

Nota

DCL - Introduzione

Definizione

Il DCL è il linguaggio per gestire i permessi e l'accesso al database.

Comandi principali:

- `CREATE USER`: crea nuovi utenti
- `DROP USER`: elimina utenti
- `GRANT`: concede privilegi
- `REVOKE`: revoca privilegi

Creazione utente:

```
CREATE USER 'mario'@'%' IDENTIFIED BY 'password123';
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin_pwd';
```

Cancellazione utente:

```
DROP USER 'mario'@'%';
```

GRANT - Concessione Privilegi

Sintassi base:

```
GRANT privilegi ON oggetto
TO utente
[WITH GRANT OPTION];
```

Esempi:

```
-- Tutti i privilegi su una tabella
GRANT ALL PRIVILEGES ON Impiegati TO 'mario'@'%';

-- Privilegi specifici
GRANT SELECT, INSERT ON Studenti TO 'utente1'@'localhost';

-- Su tutto il database
GRANT SELECT ON UniversitaDB.* TO 'lettore'@'%';

-- Con possibilit  di delegare
GRANT SELECT ON Impiegati TO 'admin'@'%'
WITH GRANT OPTION;
```

REVOKE - Revoca Privilegi

Sintassi:

```
REVOKE privilegi ON oggetto FROM utente;
```

Esempi:

```
-- Revoca tutti i privilegi
REVOKE ALL PRIVILEGES ON Impiegati FROM 'mario'@'%';

-- Revoca privilegi specifici
REVOKE INSERT, UPDATE ON Studenti FROM 'utente1'@'localhost';

-- Revoca su database
REVOKE SELECT ON UniversitaDB.* FROM 'lettore'@'%';
```

Importante

- Un utente può revocare **solo privilegi che lui ha concesso**
- La revoca non è automaticamente ricorsiva

Privilegi Disponibili

Privilegi sui dati:

- `SELECT`
- `INSERT`
- `UPDATE`
- `DELETE`

Privilegi sulla struttura:

- `CREATE`
- `ALTER`
- `DROP`
- `INDEX`

Privilegi amministrativi:

- `ALL PRIVILEGES`
- `GRANT OPTION`
- `CREATE USER`
- `RELOAD`
- `SHUTDOWN`

Privilegi speciali:

- `EXECUTE` (procedure)
- `REFERENCES`
- `TRIGGER`

Riepilogo Costrutti DDL

sqlblue!20 **Operazione**	**Comando**
Creare schema	`CREATE SCHEMA`
Creare tabella	`CREATE TABLE`
Modificare tabella	`ALTER TABLE`
Cancellare tabella	`DROP TABLE`
Creare vista	`CREATE VIEW`
Cancellare vista	`DROP VIEW`
Concedere privilegi	`GRANT`
Revocare privilegi	`REVOKE`

Ricorda

Il DDL modifica la **struttura** del database, non i dati. Le operazioni sono **permanenti** e vanno eseguite con attenzione.

Bibliografia e Riferimenti

- SQL Standard - ISO/IEC 9075
- MySQL Reference Manual - `https://dev.mysql.com/doc/`
- PostgreSQL Documentation - `https://www.postgresql.org/docs/`
- Atzeni, Ceri, Paraboschi, Torlone - "Basi di Dati"
- Elmasri, Navathe - "Fundamentals of Database Systems"

Grazie per l'attenzione!

Per domande o chiarimenti:
`massimo.tivoli@iisfermi.edu.it`