

SQL - DDL (Data Definition Language)

Linguaggio di Definizione dei Dati

Prof. Fedeli Massimo - Tutti i diritti riservati

Fabbrica Digitale

16 gennaio 2026

- 1 Introduzione a SQL
- 2 Interfacce Grafiche
- 3 Il Linguaggio SQL
- 4 Creazione di Tabelle
- 5 Tipi di Dati
- 6 Vincoli di Base
- 7 Chiavi Primarie
- 8 Vincolo UNIQUE
- 9 Politiche di reazione - Chiavi Esterne (Foreign Key)
- 10 Operazioni su Foreign Key
- 11 Inserimento Dati

- **DBMS** → “strato software” che si frappa fra l'utente e i dati veri e propri presenti nel database
- Permette di gestirli e organizzarli
- Tramite il DBMS utente e applicazioni **non** accedono direttamente al supporto fisico dove sono memorizzati i dati
- I dati vengono “visti” attraverso la loro **rappresentazione logica** (modello logico)

- Se nel ciclo di vita di un sistema informativo vengono sostituiti gli archivi fisici anche modificando la struttura dei file
- Ma mantenendo lo stesso **schema logico**
- Le applicazioni non necessitano di interventi
- In quanto agiscono a uno strato superiore di virtualizzazione

La rappresentazione logica viene anche chiamata:

- “schema del database”
- L'utente del database può interagire con esso solo tramite questa rappresentazione dei dati
- Direttamente utilizzando gli strumenti grafici messi a disposizione dal DBMS
- Mediante appositi linguaggi di programmazione (**SQL**)

La maggior parte dei moderni DBMS offre all'utente un'interfaccia grafica che semplifica le operazioni di:

- Creazione delle tabelle
- Definizione dei vincoli
- Interrogazione per individuare istanze ed estrarre elenchi
- Definizione e aggiornamento dei dati

MySQL è il database **open source** più popolare al mondo:

- Fornisce la soluzione a ogni tipo di applicazione
- Database scalabile in rete e ad alte prestazioni
- Mette a disposizione dell'utente un'interfaccia grafica ([phpMyAdmin](#))
- Consente di definire le tabelle
- Importare ed esportare i dati in molteplici formati senza scrivere alcuna riga di codice

- È “ospitato” all'interno di Apache, il Web server più utilizzato nel Web
- Integrato in svariati prodotti (XAMPP, WAMP, LAMP, ecc.)
- Disponibili altri prodotti free per la sua gestione:
 - HeidiSQL
 - MariaDB
- Offrono ulteriori possibilità sia per l'amministrazione che per la gestione dei dati

SQL - Structured Query Language

- Il linguaggio SQL (Structured Query Language) consente di interagire con i dati presenti nelle basi di dati
- Se utilizzato da solo, l'SQL non permette di programmare un'intera applicazione
- SQL **non è computazionalmente completo**, dato che non ha nel suo set di istruzioni le tre strutture fondamentali

SQL può essere suddiviso in quattro sotto linguaggi:

DDL - Data Definition Language (Linguaggio di definizione dei dati)

- È la parte del linguaggio SQL che comprende tutti i comandi preposti alla creazione di tabelle
- Definizione dei dati in esse contenuti
- Si definiscono/modificano le strutture delle relazioni dello schema della base di dati (**livello intensionale**)

DML - Data Manipulation Language

- È la parte del linguaggio contenente tutti i comandi per la **modifica dei dati** contenuti nelle tabelle
- In termini di:
 - Aggiunta
 - Eliminazione
 - Modifica
 - Aggiornamento dei dati
- Livello **estensionale**

DCL - Data Control Language

- Appartengono a questa sezione di SQL tutti i comandi che rendono possibile la **gestione dei permessi di accesso** alle risorse del database

QL - Query Language

- Rappresenta il fulcro di quasi tutte le attività legate ai database
- I comandi appartenenti a questa sezione di SQL consentono:
 - L'interrogazione
 - Il raggruppamento
 - Il conteggio
 - L'ottenimento di prospetti personalizzati dei dati presenti nelle varie tabelle dei database

Il linguaggio SQL è un **linguaggio dichiarativo**:

- Si pone a un livello di astrazione superiore rispetto ai linguaggi di programmazione tradizionali
- Utilizzabile in modalità interattiva oppure compilata
- Molte possibilità di utilizzo:
 - Linguaggi testuali interattivi (SQL)
 - Comandi simili a quelli interattivi “immersi” in un linguaggio ospite (COBOL, Java, C, ...)

CREATE TABLE - Sintassi

La creazione di una nuova relazione (tabella) viene effettuata in linguaggio SQL mediante il costrutto:

```
CREATE TABLE nomeTabella  
(  
  nomeAttributo1 tipo [valoreDefault] [vincoli],  
  nomeAttributo2 tipo [valoreDefault] [vincoli],  
  ...  
);
```

Per ciascun attributo si deve indicare:

- Il nome e il suo dominio
- Facoltativamente: valore di default e vincoli

Esempio CREATE TABLE

Vogliamo creare nel database scuola una tabella studenti per memorizzare:

- Nome e cognome dell'alunno
- Matricola
- Indirizzo

```
CREATE TABLE studenti(  
matricola CHAR(4) PRIMARY KEY,  
nome CHAR(40),  
indirizzo CHAR(80)  
);
```


Il dominio di un attributo è l'insieme dei valori che possono essere presenti in una colonna:

- **Standard:** definito implicitamente dal linguaggio come per il tipo di dato dei linguaggi di programmazione
- **Definiti dall'utente:** semplici, ma riutilizzabili

I principali tipi di dati disponibili sono raggruppati in cinque categorie:

- 1 Numerici
- 2 Stringhe di caratteri
- 3 Stringhe di bit
- 4 Date
- 5 Booleani
- 6 Immagini

I dati numerici sono di due tipi, con e senza decimali:

Numeri interi: possono essere di dimensioni differenti

- INTEGER, INT, SMALLINT, BIGINT
- È possibile indicare l'assenza di segno con la clausola UNSIGNED

Numeri reali con dimensionamento fisso:

- NUMERIC(n,d) oppure DECIMAL(n,d)
- n = numero di cifre totali
- d = cifre decimali

Esempio: NUMERIC(4.2) → numeri reali tra -99.99 e +99.99

Numeri reali a virgola mobile:

- FLOAT oppure REAL (precisione singola)
- DOUBLE (precisione doppia)

In alcune versioni di SQL è possibile definire la dimensione della mantissa e dell'esponente:

- FLOAT (5,2) → mantissa 5 cifre frazionarie, esponente 2 cifre intere
- DOUBLE (8.4) → mantissa 8 cifre frazionarie, esponente 4 cifre intere

Chiavi Artificiali - AUTO_INCREMENT

Per i numeri interi l'aggiunta della keyword `AUTO_INCREMENT` consente di creare dei campi numerici che si autoincrementano a ogni nuovo inserimento nella tabella.

Particolarmente utilizzati nelle **chiavi artificiali**:

- `INTEGER AUTO_INCREMENT`
- `SMALLINT AUTO_INCREMENT`

Stringhe di Caratteri

Le stringhe di caratteri possono essere di lunghezza fissa oppure variabile:

CHAR(n): definisce attributi con lunghezza fissa di n caratteri

- Occupa sempre una quantità fissa di spazio
- Indipendentemente dalla lunghezza effettiva della stringa

VARCHAR(n): definisce attributi con lunghezza variabile con un numero massimo n di caratteri

- Occupa solo lo spazio necessario per la lunghezza effettiva della stringa
- Nel tipo VARCHAR l'occupazione totale di memoria richiede un byte aggiuntivo utilizzato dal DBMS come prefisso

Le stringhe di bit possono essere di lunghezza fissa oppure variabile, con dimensione minima di un bit:

- `BIT(n)`: definisce attributi con lunghezza fissa di n bit
- `BIT VARYING(n)`: definisce attributi con lunghezza variabile con un numero massimo n di bit

Possono essere di differenti tipi e complessità:

- DATE: ha 10 posizioni composta da tre parti, con componenti 'AAAA-MM-GG'
- TIME: ha almeno 8 posizioni strutturato con componenti 'hh:mm:ss'
- TIMESTAMP: comprende entrambi, con componenti 'AAAAMMGGhhmmss'
 - Estremamente importante per aggiungere le marche temporali
 - Fondamentali per tutti i record che richiedono una validazione con data certa

- Il DBMS provvede automaticamente alla validazione delle date
- Accetta solo valori coerenti in base alla durata dei singoli mesi
- Riconoscendo gli anni bisestili tramite un calendario perpetuo
- Sono definite un insieme di funzioni che permettono l'agevole gestione di questi tipi di dati

Domini e tipi di dati: booleani (introdotti in SQL-3)

Anche se si potevano implementare attributi booleani semplicemente con stringhe di bit di lunghezza unitaria, SQL-3 ha introdotto come tipo semplice il BOOLEAN, che può assumere i valori TRUE e FALSE.

Domini tipi di dati per documenti di grandi dimensioni (SQL-3)

Introdotti per rappresentare oggetti di grandi dimensioni costituiti da una sequenza arbitraria di valori binari o di caratteri:

- BLOB (binary large object)
- CLOB (character large object)

Con la creazione della tabella è possibile completare la definizione degli attributi aggiungendo alcune caratteristiche essenziali per la corretta memorizzazione dei dati:

- Vincoli di chiave
- Vincoli di integrità referenziale (intrarelazionali)

I **vincoli intrarelazionali** sono quelli esistenti sui campi di un'unica relazione e devono essere rispettati da tutte le istanze di quel dominio o attributo.

Per ciascun attributo è possibile indicare quale deve essere il suo valore di DEFAULT:

- Può essere sia un valore qualunque del suo dominio
- Sia il valore NOT NULL
- Dove non specificato il valore di default è NULL

Esempio DEFAULT e AUTO_INCREMENT

```
CREATE TABLE interrogazioni(  
  ID_interrogazione INT NOT NULL AUTO_INCREMENT,  
  id_alunno INT NOT NULL,  
  data_interrogazione DATE,  
  voto INT DEFAULT 1  
)
```

- ID_Interrogazione → Intero, not null, si autoincrementa
- voto → intero con valore di default 1

È possibile limitare i valori per ogni attributo usando la clausola CHECK:

```
CREATE TABLE interrogazioni(  
  ID_interrogazione INT NOT NULL AUTO_INCREMENT,  
  id_alunno INT DEFAULT NOT NULL,  
  data_interrogazione DATE,  
  voto INT DEFAULT 1 CHECK (VALUE >= 1 AND value <= 10)  
)
```

Il voto deve essere compreso tra 1 e 10.

Primary Key - Un Attributo

L'indicazione della chiave primaria viene effettuata con la clausola PRIMARY KEY (PK):

```
CREATE TABLE interrogazioni(  
  ID_interrogazione INT NOT NULL AUTO_INCREMENT  
  PRIMARY KEY,  
  id_studente CHAR(20),  
  id_materia INT,  
  data_interrogazione DATE,  
  voto INT DEFAULT 1 CHECK (voto >= 1 AND voto <= 10)  
)
```

Primary Key - Chiave Composta

A differenza di UNIQUE e NOT NULL che possono essere definiti su più attributi della stessa tabella, il vincolo PRIMARY KEY deve essere unico nella tabella.

Nel caso di chiave composta da più attributi si utilizza la seguente notazione:

```
CREATE TABLE interrogazioni(  
  id_studente CHAR(20),  
  id_materia INT,  
  data_interrogazione DATE,  
  voto INT DEFAULT 1 CHECK (voto >= 1 AND voto <= 10),  
  PRIMARY KEY(id_studente, id_materia, data_interrogazione)  
)
```

Per gli attributi che fanno parte della chiave primaria il vincolo NOT NULL è implicito.

Aggiunta Primary Key a Tabella Esistente

È possibile aggiungere in un secondo tempo il vincolo di chiave primaria su uno o più campi mediante la clausola di modifica `ALTER`:

```
ALTER TABLE dipartimenti  
ADD PRIMARY KEY(nome_dipartimento)
```

Aggiunta Attributo a Tabella Esistente

Con la stessa clausola possiamo aggiungere attributi a una tabella:

```
ALTER TABLE docenti ADD data_nascita DATE
```

Attenzione

Lasciamo al lettore verificare la correttezza delle istruzioni nei diversi database in quanto, pur essendo uno standard, le istruzioni spesso non sono completamente compatibili e hanno lievi differenze sintattiche.

Per esempio, Access non permette di specificare la dimensione del campo numerico e con `NUMERIC` indica il reale a doppia precisione.

Il Vincolo di Unicità UNIQUE

- Per dichiarare una o più colonne i cui valori devono essere necessariamente distinti all'interno di una tabella SQL prevede il vincolo **UNIQUE**
- La clausola **unique** si applica ad un attributo o un insieme di attributi di una tabella e impone che i valori dell'attributo siano una superchiave
- Viene fatta eccezione per il valore **NULL**, il quale può comparire su diverse righe senza violare il vincolo
- In una tabella è possibile specificare più chiavi **UNIQUE** ma una sola **PRIMARY KEY**

UNIQUE - Esempio 1

```
CREATE TABLE Studenti (  
  ID INT PRIMARY KEY,  
  Nome VARCHAR(50),  
  Cognome VARCHAR(50),  
  DataNascita DATE,  
  UNIQUE(Nome, Cognome)  
);
```

Le coppie Nome e Cognome devono avere sempre valori diversi.

UNIQUE - Esempio 2

```
CREATE TABLE Studenti (  
  ID INT PRIMARY KEY,  
  Nome VARCHAR(50) UNIQUE,  
  Cognome VARCHAR(50) UNIQUE,  
  DataNascita DATE  
);
```

In questo caso si impone che nome e cognome singolarmente abbiano sempre valori diversi.

Politiche di reazione

I vincoli interrelazionali sono quelli esistenti tra due differenti tabelle (relazioni) e riguardano le chiavi esterne.

Due tabelle (relazioni) si chiamano rispettivamente:

- **Esterna:** tabella contenente l'insieme delle chiavi, dove è definito l'attributo
 - Tabella che ha esistenza propria (es. anagrafica dipendenti)
- **Interna:** tabella contenente la chiave Foreign Key di collegamento alla tabella esterna, che prende appunto il nome di chiave esterna

Foreign Key - Definizione

- La Foreign Key crea un legame tra i valori di un attributo della tabella interna e i valori di un attributo di un'altra tabella esterna
- Il vincolo impone che per ogni riga della tabella, il valore dell'attributo specificato, se diverso dal valore nullo, sia presente nelle righe della tabella esterna (principale)
- L'unico requisito che la sintassi impone è che l'attributo cui si fa riferimento nella tabella esterna sia soggetto a un vincolo `UNIQUE`
- Tipicamente l'attributo della tabella esterna fa parte della chiave primaria

Ci sono due notazioni che permettono di definire questo vincolo e utilizzano le clausole REFERENCES e FOREIGN KEY:

- A seconda del caso, il vincolo di integrità referenziale riguarda uno o più di un attributo delle tabelle interne
- Per impostare le politiche di reazione su un database esistente, è necessario prima individuare le tabelle esterne (quelle con esistenza propria) e le tabelle interne ovvero quelle con le chiavi esterne.
- le politiche di reazione devono essere inserite nelle tabelle interne.

Foreign Key - Esempio 1

Mettiamo in relazione i **dipartimenti** (tabella interna) con i **docenti** (esterna):

```
CREATE TABLE dipartimenti(  
  nome_dipartimento CHAR(15) PRIMARY KEY  
)
```

```
CREATE TABLE docenti(  
  ID_docente INT NOT NULL PRIMARY KEY,  
  nome CHAR(20) NOT NULL,  
  cognome CHAR(20) NOT NULL,  
  dipartimento CHAR(15)  
  REFERENCES dipartimenti(nome_dipartimento)  
)
```

Foreign Key - Esempio 2 (Chiave Composta)

Relazione tra **studenti** (interna) e **corsi** (esterna):

```
CREATE TABLE corsi(  
  classe INT,  
  sezione CHAR(4),  
  nome_corso CHAR(25),  
  PRIMARY KEY(classe, sezione)  
)
```

```
CREATE TABLE studenti(  
  matricola CHAR(20) PRIMARY KEY,  
  nome VARCHAR(20), cognome VARCHAR(20),  
  data_nascita DATE,  
  classe INT, sez CHAR(4),  
  FOREIGN KEY(classe, sez)  
  REFERENCES corsi(classe, sez)  
)
```

Nel ciclo di vita del database potrebbero essere effettuate azioni sui dati di una tabella esterna:

- I dati potrebbero essere modificati (UPDATE)
- Potrebbe essere cancellato l'intero record (DELETE)

Esempio: Se venisse cancellato il dipartimento di fisica, quale conseguenza ne deriverebbe per la chiave esterna presente nella tabella docenti?

Nel DBMS è possibile impostare un'operazione alternativa da effettuare in caso di violazione del vincolo di integrità referenziale nella tabella interna:

- SET NULL: pone uguale a NULL la chiave esterna
- CASCADE: esegue la medesima operazione (UPDATE/DELETE) sui record connessi
- SET DEFAULT: ripristina i valori di default
- NO ACTION: nessuna azione

Esempio con Operazioni Alternative

```
CREATE TABLE docenti(  
  ID_docente INT NOT NULL PRIMARY KEY,  
  nome CHAR(20) NOT NULL,  
  cognome CHAR(20) NOT NULL,  
  dipartimento CHAR(15)  
  REFERENCES dipartimenti(nome_dipartimento)  
  ON DELETE SET DEFAULT  
  ON UPDATE CASCADE  
)
```

Esempio Studenti con FK

```
CREATE TABLE studenti(  
  matricola CHAR(20) PRIMARY KEY,  
  nome VARCHAR(20),  
  cognome VARCHAR(20),  
  data_nascita DATE,  
  classe INT,  
  sez CHAR(4),  
  FOREIGN KEY(classe, sez)  
  REFERENCES corsi(classe, sez)  
  ON DELETE SET DEFAULT  
  ON UPDATE SET DEFAULT  
)
```

A) Cancellazione con vincolo SET NULL

Effettuando la cancellazione del record *Fisica* nella tabella *dipartimenti* automaticamente il DBMS provvede ad aggiornare tutti i riferimenti a esso presenti nelle chiavi esterne sostituendo in essi il valore NULL.

B) Cancellazione/modifica con vincolo CASCADE

In questo caso effettuando la cancellazione del record *Fisica* nella tabella *dipartimenti* automaticamente il DBMS provvede a cancellare tutti i record che hanno una chiave esterna che contiene il valore rimosso.

Analogamente, se venisse modificato un valore nella chiave della tabella esterna, la stessa modifica verrebbe effettuata in tutte le sue occorrenze.

C) Cancellazione con vincolo SET DEFAULT

Effettuando la cancellazione del record *Fisica* nella tabella *dipartimenti*, il DBMS provvede a inserire in tutti i record che hanno una chiave esterna che contiene il valore rimosso un valore definito all'atto della creazione della tabella:

- Per gli attributi alfanumerici spesso è il campo vuoto, cioè ""

D) Cancellazione con vincolo NO ACTION

In questo caso non vengono fatti controlli sulle altre tabelle, lasciandole invariate, quindi la cancellazione del record *Fisica* nella tabella *dipartimenti* non provoca modifiche nella tabella interna.

Riepilogo Operazioni su FK

Le opzioni ON DELETE e ON UPDATE impongono ai DBMS le azioni seguenti:

| Opzione | Azione |
|-------------|---|
| CASCADE | Propagare in “cascata” (CASCADE) le cancellazioni (DELETE) oppure le modifiche (UPDATE) anche alla tabella dal lato molti |
| NO ACTION | Impedire il tentativo di cancellazione o modifica annullando l'intera operazione |
| SET NULL | Assegnare alla chiave esterna il valore NULL |
| SET DEFAULT | (se si usa SET NULL) oppure il valore di default (se si usa SET DEFAULT) |

In genere, nei DBMS l'opzione NO ACTION è quella di default, accompagnata da un messaggio di errore per l'utente che segnala il tentativo di violazione dell'integrità referenziale.

Inserimento di Dati con Riferimenti Mancanti

- Come per le operazioni di cancellazione e di modifica, anche le operazioni di inserimento di dati in tabelle tra loro connesse sono soggette a vincoli referenziali
- Un dato che deve essere inserito potrebbe non essere presente nella tabella esterna
- L'impostazione di questi vincoli viene ricondotta quindi a quelle di UPDATE
- Considerando gli effetti dell'inserimento di un nuovo attributo equivalenti a quelli della variazione di un attributo presente in uno mancante

- ① **Inserzione dipendente:** consente l'inserzione di un'istanza nella tabella interna solo se la chiave nella tabella esterna esiste già
- ② **Inserzione automatica:** viene effettuata l'inserzione di un'istanza nella tabella interna e, nel caso in cui l'istanza nella tabella esterna non esista, viene creata
- ③ **Inserzione nulla:** viene effettuata l'inserzione di una tupla nella tabella interna e, nel caso che l'istanza nella tabella esterna non esista, la fk nella tabella interna viene messa a NULL

- 4 **Inserzione di default:** viene effettuata l'inserzione di un'istanza dell'entità interna e, se l'istanza dell'entità esterna non esiste, la fk della tabella interna viene impostata a un valore predefinito
- 5 **Nessun effetto:** con questa regola è sempre permessa l'inserzione di un'istanza nella tabella interna e non è richiesta in alcun modo l'esistenza nella tabella esterna

Grazie per l'attenzione!

Per domande o chiarimenti:
Prof. Massimo
IIS Fermi Sacconi Ceci - Ascoli Piceno