Basi di dati

Il Linguaggio SQL

Data Definition Language (DDL)

Data Definition Language:
insieme di istruzioni utilizzate per modificare la struttura della base di dati

Ne fanno parte le istruzioni di **inserimento, cancellazione** e **modifica** di tabelle.

Il **DDL** di SQL permette:

- di definire schemi di relazioni (o “table”, tabelle), modificarli ed eliminarli
- di specificare vincoli, sia a livello di tupla (o “riga”) che a livello di tabella
- di definire nuovi domini, oltre a quelli predefiniti
- definizione delle viste (“view”), ovvero tabelle virtuali

Creazione di una tabella

La creazione di una tabella avviene attraverso l'enumerazione delle colonne che la compongono.

- Mediante l'istruzione **CREATE TABLE** si definisce lo schema di una tabella e se ne crea un'istanza vuota.
- Per ogni attributo và specificato il dominio, un eventuale valore di default e eventuali vincoli.

CREATE TABLE Tabella
(<nome_campo1> <tipo_campo1>,
<nome_campo2> <tipo_campo2>,...)

```
CREATE TABLE Studenti(
    Matricola char(10),
    Nome char(20) NOT NULL,
    Cognome char(20) NOT NULL,
    AnnoDiIscrizione integer,
    CONSTRAINT "PK_Studente" PRIMARY KEY (Matricola))
```

Valori di default

I valori di default specificano cosa deve essere assegnato all'attributo (colonna) quando non si indica un valore esplicitamente.

Esempi:

AnnoDiIscrizione integer **DEFAULT** 1

NumeroPatente char(20) **DEFAULT** **NULL**

Vincoli di integrità intra-relazionali

Operano sugli attributi delle relazioni:

- **NOT NULL**
- **UNIQUE** definisce chiavi
- **PRIMARY KEY**: chiave primaria (una sola, implica NOT NULL)
- **CHECK**: vincolo di controllo

Vincoli di integrità intra-relazionali

Vincolo NOT NULL:

- vieta la presenza di valori nulli

```
CREATE TABLE Studenti(
    Matricola char(10),
    Nome char(20) NOT NULL,
    Cognome char(20) NOT NULL,
    AnnoDiIscrizione integer,
    …
    PRIMARY KEY (Matricola))
```

Vincoli di integrità intra-relazionali

Vincolo UNIQUE:

➢ Non possono esistere due righe che hanno gli stessi valori per l'attributo o insieme di attributi specificati

```
CREATE TABLE Studenti(
    Matricola char(10),
    CodiceFiscale char(16) UNIQUE,
    Nome char(20) NOT NULL,
    Cognome char(20) NOT NULL,
    AnnoDiIscrizione integer,
    …
    PRIMARY KEY (Matricola))
```

Vincoli di integrità intra-relazionali

Vincolo PRIMARY KEY:

- Identifica la chiave primaria

```
CREATE TABLE Veicoli(
    Targa char(10),
    CodiceProprietario char(20) NOT NULL,
    …
    PRIMARY KEY (Targa,CodiceProprietario ))
```

Oppure

```
CREATE TABLE Veicoli(
    Targa char(10),
    CodiceProprietario char(20) NOT NULL,
    …
    CONSTRAINT "PK_Veicoli" PRIMARY KEY (Targa,CodiceProprietario ))
```

Vincoli di integrità intra-relazionali

Vincolo di controllo CHECK:

- I vincoli di controllo sono utilizzati per verificare generiche condizioni sui valori di una colonna.
- Il vincolo è violato se esiste almeno una tupla che rende falsa la condizione

```
CREATE TABLE Esami(
        Matricola char(10),
        Corso char(20) UNIQUE,
        Voto integer CHECK (Voto > 18 AND Voto < 30),
        …
        PRIMARY KEY (Corso, Matricola))
```

NOTA: Se **CHECK** viene espresso a livello di tabella (anziché nella definizione dell'attributo) è possibile fare riferimento a più attributi della tabella stessa
Es: **CHECK** (ImportoLordo = Netto + Ritenute)

Vincoli di integrità inter-relazionali

- Permettono di definire vincoli di integrità referenziale
- Creano un legame tra i valori dell'attributo di una tabella e i valori di un attributo di un'altra tabella
  - **REFERENCES** permette di specificare vincoli di colonna
  - **FOREIGN KEY** vincolo di chiave esterna

Vincoli di integrità inter-relazionali

Vincolo REFERENCES:
Permette di specificare vincoli di colonna.

```
CREATE TABLE Impiegati (
  Matricola          char(6) PRIMARY KEY,
  Cognome            varchar(50) NOT NULL,
  Nome               varchar(50) NOT NULL,
  Dipartimento       char(15) REFERENCES Dipartimenti(NomeDipartimento),
...
)
```

Il campo Dipartimento può assumere solo i valori che compaiono nel campo NomeDipartimento della tabella Dipartimenti.

Vincoli di integrità inter-relazionali

Vincolo FOREIGN KEY:
La definizione di una foreign key avviene specificando un vincolo e indicando quale chiave viene referenziata.

```
CREATE TABLE Esami(
        Matricola char(10),
        Corso char(20) UNIQUE,
        Voto integer,
        …
        PRIMARY KEY (Corso, Matricola),
        FOREIGN KEY (Matricola) REFERENCES Studenti (Matricola) )
```

Viste

Sono **tabelle virtuali** ovvero che non esistono fisicamente ma il cui contenuto è ottenuto da altre tabelle.

In SQL si ottengono assegnando una lista di attributi ed un nome ad una interrogazione con select.

Studenti(Matricola,Cognome,Nome,CittaResidenza,Media)
Esami(Matricola,Corso,Voto)

```
CREATE VIEW [OR REPLACE] StudentiMeritevoli (Matricola,Nome,Cognome)
AS SELECT Matricola,Nome,Cognome
     FROM Studenti
     WHERE Matricola IN (SELECT Matricola
                         FROM Esami
                         GROUP BY Matricola
                         HAVING AVG(Voto)>=30)
```

Viste

A cosa servono le viste?

- Dare la visibilità di tabelle non più presenti nel database
- Semplificare o potenziare la gestione dei permessi.

  Ad esempio si potrebbe creare una query che legge solo alcuni dati da una tabella (tramite la clausola WHERE), per poi assegnare il permesso in lettura ad un certo utente sulla vista, ma non sulla tabella di base. In questo modo l'utente non vedrà i dati che non vengono estratti dalla vista.

Si possono aggiornare le viste?

- Dipende dal DBMS
- Lo standard dice che una vista basata su una sola tabella debba essere aggiornabile, una basata più tabelle non deve essere

Modifica dello Schema

Istruzione **DROP TABLE**:

- Rimuove la tabella e eventualmente le tabelle dipendenti

- **CASCADE**: Elimina la tabella Studenti, il suo contenuto e le viste connesse

DROP TABLE Studenti **CASCADE**

- **RESTRICT**: Elimina la tabella Studenti solo se è vuota e non ci sono oggetti connessi

DROP TABLE Studenti **RESTRICT**

Modifica dello Schema

Istruzione **ALTER TABLE**:

- permette di inserire/rimuovere colonne dalle tabelle
- inserire/rimuovere vincoli

Inserimento della colonna Sesso nella Tabella Studenti:

```
ALTER TABLE Studenti
ADD COLUMN Sesso char(1) CHECK (Sesso in ('M','F')),
DROP CONSTRAINT Mail,
…
```

Cancellazione colonna AnnoIscrizione nella Tabella Studenti:

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

Data Manipulation Language (DML)

Il **Data Manipulation Language** è l'insieme di istruzioni utilizzate per modificare il contenuto della base di dati.

Ne fanno parte le istruzioni di inserimento, cancellazione e modifica dei record (INSERT, DELETE, UPDATE):

SELECT esegue interrogazioni (query) sul DB
INSERT INTO permette di inserire nuove tuple nel DB
DELETE permette di cancellare tuple dal DB
UPDATE permette di modificare tuple del DB

- **INSERT INTO** può usare il risultato di una query per eseguire inserimenti multipli

- **DELETE** e **UPDATE** possono fare uso di condizioni per specificare le tuple da cancellare o modificare

- In ogni caso gli aggiornamenti riguardano una sola tabella

Inserimento: INSERT INTO

È possibile inserire una nuova tupla specificandone i valori

INSERT INTO Tabella [(Campo1, Campo2....)]
VALUES (Val1, Val2,...)

INSERT INTO Studenti(Matricola,Cognome,Nome,CittaResidenza,Media)
VALUES (‘M0004’,‘Rossi’,’Paola’,‘Pisa’,24)

- Ci deve essere corrispondenza tra attributi e valori
- La lista degli attributi si può omettere, nel qual caso vale l’ordine con cui sono stati definiti
- Se la lista non include tutti gli attributi, i restanti assumono valore NULL (se ammesso) o il valore di default (se specificato)

INSERT INTO Studenti(Matricola,Cognome,Nome,CittaResidenza)
VALUES (‘M0004’,‘Rossi’,’Paola’,‘Pisa’)

Inserimento: INSERT INTO

È possibile anche inserire le tuple che risultano da una query:

```
INSERT INTO ResidenzaPisa(ResidenzaPI,Matricola)
        SELECT CittaResidenza, Matricola
        FROM Studenti
        WHERE CittaResidenza = 'Pisa'
```

- Valgono le regole viste per il caso singolo

- Gli schemi del risultato e della tabella in cui si inseriscono le tuple possono essere diversi, l'importante è che i tipi delle colonne siano compatibili

Modifica: UPDATE

L'istruzione **UPDATE** modifica i valori delle colonne in una riga esistente
- può fare uso di una condizione per specificare le tuple da modificare

UPDATE Tabella
SET <campo1>=<valore1>, <campo2>=<valore2> ....
WHERE condizione

Studenti(Matricola,Cognome,Nome,CittaResidenza,Media)
Esami(Matricola,Corso,Voto)

UPDATE Studenti **SET** Media = Media+1 **WHERE** Matricola = 'M0004'

UPDATE Studenti **SET** Media = Media-1 **WHERE** Media>25

UPDATE Esami **SET** Voto = 30
WHERE Matricola = ' M0004' **AND** corso = 'Analisi I'

L'istruzione UPDATE può portare a violare il vincolo di integrità referenziale

Cancellazione: DELETE

L'istruzione **DELETE** cancella una tupla esistente
- può fare uso di una condizione per specificare le tuple da cancellare

DELETE FROM Tabella WHERE condizione

DELETE FROM Studenti **WHERE** Matricola = 'A002038'

Cancella gli studenti che non hanno fatto esami:

```
DELETE FROM Studenti
WHERE Matricola NOT IN (SELECT Matricola
                        FROM Esami
                        GROUP BY Matricola)
```

OSS: Se la clausola WHERE non e' presente si eliminano tutti i record della tabella

L'istruzione DELETE può portare a violare il vincolo di integrità referenziale

Politiche di “reazione”

Per garantire che a fronte di cancellazioni e modifiche i vincoli di integrità referenziale siano rispettati, si possono specificare opportune **politiche di reazione** in fase di definizione degli schemi

```
CREATE TABLE Iscrizioni(
        CodIscrizione char(20),
        Matricola char(10),
        ...
        CONSTRAINT "PK_Iscrizioni" PRIMARY KEY (CodIscrizione, Matricola),
        CONSTRAINT “FK_Studenti" FOREIGN KEY (Matricola) REFERENCES Studenti
        ON DELETE CASCADE              -- cancellazione in cascata
        ON UPDATE NO ACTION            -- modifiche non permesse
)
```

- Se una riga di Studenti e’ cancellata verranno cancellate tutte le righe di Iscrizioni che la referenziano
- Se la colonna Matricola di Studenti e’ modificata, l’aggiornamento deve essere rifiutato se una riga di Iscrizioni punta alla riga modificata

Politiche di “reazione”

Sintassi:
ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }
ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }

ON DELETE NO ACTION:
Specifica che se si tenta di eliminare una riga contenente una chiave a cui fanno riferimento chiavi esterne in righe esistenti in altre tabelle, verrà generato un errore.

ON UPDATE NO ACTION:
Specifica che se si tenta di aggiornare un valore di chiave in una riga e alla chiave fanno riferimento chiavi esterne in righe esistenti in altre tabelle, verrà generato un errore.

Politiche di “reazione”

Sintassi:
ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }
ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }

ON DELETE CASCADE:
Specifica che se si tenta di eliminare una riga contenente una chiave a cui fanno riferimento chiavi esterne in righe esistenti in altre tabelle, verranno inoltre eliminate tutte le righe contenenti tali chiavi esterne.

ON UPDATE CASCADE:
Specifica che se si tenta di aggiornare un valore di chiave in una riga e a tale valore fanno riferimento chiavi esterne in righe esistenti in altre tabelle, tutti i valori che compongono la chiave esterna verranno anch'essi aggiornati al nuovo valore specificato per la chiave.

Politiche di “reazione”

Sintassi:
ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }
ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }

ON DELETE SET NULL:
Specifica che se si tenta di eliminare una riga contenente una chiave a cui fanno riferimento chiavi esterne in righe esistenti in altre tabelle, tutti i valori che compongono la chiave esterna presenti nelle righe a cui si fa riferimento verranno impostati su NULL.

NOTA:
Per l'esecuzione di questo vincolo è necessario che le colonne chiave esterna ammettano i valori Null.

Politiche di “reazione”

Sintassi:
ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }
ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }

ON UPDATE SET NULL:
Specifica che se si tenta di aggiornare una riga contenente una chiave a cui fanno riferimento chiavi esterne in righe esistenti in altre tabelle, tutti i valori che compongono la chiave esterna presenti nelle righe a cui si fa riferimento verranno impostati su NULL.

NOTA:
Per l'esecuzione di questo vincolo è necessario che le colonne chiave esterna ammettano i valori Null.

Politiche di “reazione”

Sintassi:
ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }
ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }

ON DELETE SET DEFAULT:
Specifica che se si tenta di eliminare una riga contenente una chiave a cui fanno riferimento chiavi esterne in righe esistenti in altre tabelle, tutti i valori che compongono la chiave esterna presenti nelle righe a cui si fa riferimento verranno impostati sul relativo valore predefinito.

NOTE:
In generale per l'esecuzione di questo vincolo è necessario che per tutte le colonne chiave esterna della tabella di destinazione sia disponibile una definizione predefinita.

Data Control Language (DCL)

- Il **Data Contro Language** è un linguaggio che permette di fornire o revocare agli utenti i permessi necessari per poter utilizzare i comandi di DML e DDL oltre agli stessi comandi DCL.

- All'atto della creazione di un database l'utente creatore ne diventa il proprietario. Gli altri utenti possono leggere le informazioni ma non modificare il DB.

Per modificare i diritti di accesso è possibile utilizzare i comandi:

- **GRANT**
- **REVOKE**

Data Control Language (DCL)

Concessione di privilegi:
Il comando Grant fornisce uno o più permessi ad un determinato utente su un determinato oggetto del database

GRANT ALL PRIVILEGES ON DATABASE Impiegati
TO User1 [**WITH GRANT OPTION**]

- **WITH GRANT OPTION** specifica se il privilegio può essere trasmesso ad altri utenti

Revoca di privilegi:
- Il comando Revoke revoca uno o più permessi ad un determinato utente su un determinato tipo di oggetti.
- Un utente può revocare solo privilegi che lui ha concesso

REVOKE ALL PRIVILEGES ON Impiegati **FROM** User1