Il linguaggio SQL costrutti per la definizione delle tabelle

c.vallati@iet.unipi.it

Sommario

- Costrutti per la creazione delle tabelle e modifica della loro struttura
- Costrutti per l'inserimento dei dati
- Costrutti per il controllo degli accessi

Data Definition Language (DDL)

- Insieme di istruzioni utilizzate per modificare la struttura della base di dati, tra cui 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
  - di **definire le viste** (“**view**”), ovvero *tabelle virtuali*

Creazione di uno schema

- SQL consente la definizione di uno schema di base di dati come collezione di tabelle attraverso la seguente sintassi:

CREATE SCHEMA NomeSchema [autorizzazione];

- Creato un determinato schema questo può essere selezionato per le future azioni attraverso il seguente comando:

USE NomeSchema;

- Per cancellare un intero database invece:

DROP SCHEMA NomeSchema;

Creazione di una tabella

- La creazione di una tabella avviene attraverso l'enumerazione delle colonne che la compongono. Per ogni attributo và specificato il dominio, un eventuale valore di default e eventuali vincoli.
- Mediante l'istruzione **CREATE TABLE** si definisce lo schema di una tabella e se ne crea un'istanza vuota.
- I tipi di dato associati con ciascun attributo possono essere scelti tra i tipi di base già definiti in SQL oppure tra nuovi tipi definiti dall'utente

Sintassi:

CREATE TABLE Tabella

(<nome_campo1> <tipo_campo1>,

<nome_campo2> <tipo_campo2>,...)

Esempio:

```
CREATE TABLE Studenti(
    Matricola char(10),
    Nome char(20) NOT NULL,
    Cognome char(20) NOT NULL,
    AnnoDiIscrizione integer,
    CONSTRAINT pk_matricola 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 d’integrità

- Durante la creazione di una tabella, possono essere definiti i vincoli d’integrità
- Questi (a seconda del tipo) possono essere associati ad un attributo singolo o a tutta la relazione
- Le tipologie di vincoli sono i seguenti:
  - Vincoli intra-relazionali
  - Vincoli inter-relazionali
  - Vincoli controllo

Vincoli 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 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))
```

- *Il vincolo unique impone che gli attributi formino una super-chiave*

Vincoli intra-relazionali

Vincolo PRIMARY KEY:

- Identifica la chiave primaria

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

Le stesse tre modalità possono essere usate per UNIQUE

Oppure

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

Oppure

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

Solo nel caso in cui la chiave primaria sia composta da un solo attributo.

Vincoli 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 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

Vincolo REFERENCES

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, il campo che viene referenziato, in questo caso NomeDipartimento, non è chiave

Vincolo FOREIGN KEY

- **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 secondo la seguente sintassi:

CREATE VIEW NomeVista [ListaAttributi] AS SelectSQL

Esempio:

- Creare una vista impiegati costosi che hanno uno stipendio anunale maggiore di 30000 all'interno del database azienda:

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

Viste

- Una volta creata la vista questa può essere interrogata come una tabella normale:

  SELECT * FROM ImpiegatiCostosi;

- Altro esempio, creare una vista ImpiegatiRecenti per recuperare gli impiegati assunti dopo il 2005:

CREATE VIEW ImpiegatiRecenti
(Matricola,Nome,Cognome)

AS SELECT Matricola,Nome,Cognome

FROM impiegati

WHERE DataAssunzione > ‘2005-01-01’

Modifica dello schema

- Istruzione **DROP TABLE**:
  - Rimuove la tabella e eventualmente le tabelle dipendenti
- **Esistono due modalità di drop (anche se oggi sono in disuso):**
  - **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**
- Sintassi:
  - **DROP table NomeElemento [RESTRICT | CASCADE]**
  - La stessa sintassi può essere usata per effettuare il drop di schemi, domini, o viste usando le rispettive parole chiave schema, domain, view

Modifica dello schema

- Istruzione **ALTER TABLE**:
  - permette di inserire/rimuovere colonne dalle tabelle
  - inserire/rimuovere vincoli
- La sintassi è la seguente:
  - **ALTER TABLE NomeTabella {ADD COLUMN NomeAttributo | DROP COLUMN NomeAttributo}**

Esempi:

- Inserimento della colonna Sesso nella Tabella Studenti:
  - **ALTER TABLE** Studenti **ADD COLUMN** Sesso char(1) **CHECK** (Sesso in ('M','F')),
- Cancellazione colonna AnnoIscrizione nella Tabella Studenti:
  - **ALTER TABLE** Studenti **DROP COLUMN** AnnoIscrizione

Modifica dello schema

- Rimuovere un vincolo
  - **ALTER TABLE** nome_tabella **DROP INDEX** nome_vincolo
  - Se non si è definito un vincolo questo non può essere rimosso

```
CREATE TABLE Esami(
….
CONSTRAINT pk PRIMARY KEY (Corso, Matricola),
CONSTRAINT fk FOREIGN KEY (Matricola) REFERENCES Studenti (Matricola)
)

ALTER TABLE Esami DROP INDEX fk;
```

Data Manipulation Language

- 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

- È possibile inserire una nuova tupla specificandone i valori usando la seguente sintassi:
- **INSERT INTO Tabella [(Campo1, Campo2....)]**
- **VALUES (Val1, Val2,...)**

Esempio:

- **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

INSERT INTO VALUES ('M0004','Rossi','Paola','Pisa'),
('M0005','Verdi','Veronica','Roma')

- Se la lista non include tutti gli attributi, i restanti assumono valore NULL (se ammesso) o il valore di default (se specificato)

Inserimento

- 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

- 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

Esempio:

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, in tal caso l'inserimento fallisce.

Cancellazione

- L'istruzione **DELETE** cancella una tupla esistente
- Può fare uso di una condizione per specificare le tuple da cancellare attraverso la seguente sintassi:

DELETE FROM Tabella WHERE condizione

Esempio:

- Cancella uno studente con una matricola specifica:

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)

- NOTA: 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**

Vincoli interrelazionali update e cancellazioni

- 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
- In particolare tramite la seguente sintassi può essere definita l'azione che il sistema intraprende quando viene richiesta la cancellazione o l'update di una tupla con un attributo legato da vincolo interrelazionale:

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

NO ACTION

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.

Vincoli interrelazionali update e cancellazioni

CASCADE

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

Vincoli interrelazionali update e cancellazioni

SET NULL

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.

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.

Vincoli interrelazionali update e cancellazioni

SET DEFAULT

ON UPDATE SET DEFAULT

- 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 sul relativo valore predefinito.

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.

Vincoli interrelazionali update e cancellazioni

Esempio:

```
CREATE TABLE Iscrizioni(
        CodIscrizione char(20),
        Matricola char(10),
        ...
        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
)
```

- 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

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)

- Creazione di nuovi utenti

CREATE USER ‘Nome'@‘%' IDENTIFIED BY ‘password';

- Cancellazione di utenti

DROP USER ‘Nome’

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. Il comando ha la seguente sintassi:

GRANT ALL PRIVILEGES ON 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