Azzolini Riccardo 2019-10-03

Linguaggio SQL: comandi del DDL

1 Linguaggio SQL

Il linguaggio per la definizione e la manipolazione dei dati supportato da tutti i DBMS relazionali è l'**SQL** (*Structured Query Language*). Esso è diventato uno standard ufficiale nel 1986, e ha poi subito varie revisioni (la più recente è SQL:2016), una delle quali (SQL:1999/SQL3) introdotto delle caratteristiche *object-relational*. In realtà, le versioni di SQL implementate dai DBMS presentano alcune piccole variazioni rispetto allo standard.

L'SQL è un linguaggio dichiarativo:

- descrive *cosa fare*, e non come farlo (che è invece compito del DBMS): il risultato che si vuole ottenere da un operazione è specificato mediante condizioni sul contenuto dei dati;
- si pone a un livello di astrazione superiore rispetto ai linguaggi di programmazione tradizionali (procedurali);
- è basato sull'algebra relazionale (in particolare, per essere eseguita, ogni query SQL viene decomposta dal DBMS in tante "piccole" operazioni dell'algebra relazionale).

Esso è sia DDL che DML: comprende istruzioni per la definizione, l'interrogazione e l'aggiornamento dei dati.

Operazione	DDL (schema)	DML (istanza)
Creazione	`CREATE`	`INSERT`
Modifica	`ALTER`	`UPDATE`
Cancellazione	`DROP`	`DELETE`
Interrogazione		`SELECT`

2 Notazione

Per descrivere la sintassi di SQL, viene qui usata la seguente notazione:

- parole chiave del linguaggio: caratteri maiuscoli
- termini variabili: tra `< >`
- componenti opzionali: tra `[ ]`

- 0 o più occorrenze: `*`
- opzioni mutuamente esclusive: tra `{ }`, separate da `|`

3 Tipi di dato

I tipi di dato in SQL si suddividono in:

- tipi *predefiniti*;
- tipi *user-defined* (ad esempio tipi composti), che sono parte delle caratteristiche object-relational di SQL.

I principali tipi predefiniti sono:

- **tipi carattere**: singoli caratteri o stringhe, anche di lunghezza variabile;
- **tipi numerici**, esatti e approssimati;
- **tipi temporali**: date, ore e intervalli di tempo;
- **booleani**;
- **BLOB** (Binary Large Object) e **CLOB** (Character Large Object), rispettivamente per dati binari (es. immagini) e testi di grandi dimensioni.

4 Creazione di relazioni

La sintassi di base per la creazione di una relazione/tabella è:

```
CREATE TABLE <nome relazione>
(<specifica colonna> [, <specifica colonna>]*);
```

dove:

- `<nome relazione>` è il nome della relazione che viene creata;
- `<specifica colonna>` è composta da:

  `<nome colonna> <dominio> [DEFAULT <valore default>]`

  - `<nome colonna>` è il nome della colonna, che deve essere distinto da quelli di altre colonne della stessa relazione;
  - `<dominio>` è il dominio della colonna, scelto tra i tipi di dato SQL;

– `valore default` è un valore appartenente al dominio, assunto dalle tuple se queste vengono inserite senza specificare alcun valore per la colonna (in assenza di un valore di default, viene usato `NULL`, se questo è ammesso nella colonna, altrimenti il DBMS rifiuta l'inserimento e segnala un errore).

4.1 Esempio

```
CREATE TABLE Video (
  colloc DECIMAL(4),
  titolo VARCHAR(30),
  regista VARCHAR(20),
  tipo CHAR DEFAULT 'd'
);
```

5 Correttezza dei dati

I dati contenuti in una base di dati sono corretti se soddisfano un insieme di regole di correttezza. Le operazioni di modifica dei dati definiscono un nuovo stato della base di dati, che non è necessariamente corretto. Per questo, quando si effettuano queste operazioni bisogna verificare la correttezza del nuovo stato. Tale verifica può essere effettuata:

- *dalle procedure applicative*;
- dal DBMS, mediante la definizione di *vincoli di integrità*.

5.1 Procedure applicative

Vantaggi:

- Effettuare tutte le verifiche necessarie all'interno di ogni applicazione è un approccio molto efficiente (perché non serve mandare i dati al DBMS e attenderene la risposta per sapere se essi siano validi o meno).

Svantaggi:

- È possibile aggirare le verifiche interagendo direttamente con il DBMS.
- Soprattutto per controlli complessi, ci potrebbero essere degli errori di codifica, e in questo caso tali controlli non verrebbero effettuati correttamente.

- I controlli devono essere implementati separatamente in tutti i casi in cui si effettuano modifiche alla base di dati. Siccome, però, la conoscenza delle regole di correttezza è tipicamente "nascosta" nelle applicazioni esistenti, alcuni controlli potrebbero essere dimenticati in eventuali nuove applicazioni (soprattutto se sviluppate da programmatori/team diversi).

5.2 Vincoli di integrità

I vincoli di integrità si definiscono mediante le istruzioni di creazione/modifica delle relazioni (`CREATE` e `ALTER TABLE`). In seguito, durante l'esecuzione di qualsiasi operazione di modifica dei dati, il DBMS verifica automaticamente che i vincoli siano rispettati: in caso contrario, esso rifiuta tale operazione e genera un errore di esecuzione.

Vantaggi:

- I vincoli si definiscono in modo dichiarativo, sfruttando il linguaggio SQL, mentre la verifica è affidata al sistema.
- Si ha un unico punto centralizzato di verifica, quindi è impossibile aggirare i vincoli.

Svantaggi:

- La necessità di inviare i dati al DBMS e attendere la risposta di quest'ultimo può rallentare l'esecuzione delle applicazioni.[1]
- Non è possibile definire tipologie arbitrarie di vincoli: si è limitati ai tipi supportati dal DBMS.

6 Tipi di vincoli di integrità

In SQL, quando si definisce una relazione, è possibile specificare diversi tipi di VI:

- obbligatorietà di colonne (`NOT NULL`);
- chiavi (`PRIMARY KEY` e `UNIQUE`);
- chiavi esterne (`FOREIGN KEY`);
- vincoli `CHECK`, che sfruttano il linguaggio di interrogazione.

[1] Una possibile soluzione, utile soprattutto per i programmi di data entry, consiste nell'implementare i controlli semplici sia nelle applicazioni che mediante vincoli di integrità, migliorando così i tempi di risposta nel caso degli errori più comuni e facili da individuare.

7 Obbligatorietà di colonne

Per specificare che una colonna è obbligatoria, cioè che non può assumere valori nulli, è sufficiente includere `NOT NULL` nella specifica di tale colonna.

7.1 Esempio

```
CREATE TABLE Video (
  colloc DECIMAL(4),
  titolo VARCHAR(30) NOT NULL,
  regista VARCHAR(20) NOT NULL,
  tipo CHAR DEFAULT 'd'
);
```

8 Chiavi

Il linguaggio SQL permette di specificare due tipi di chiavi:

- `PRIMARY KEY` impone che, per ogni tupla, i valori degli attributi che compongono la chiave siano *non nulli* e *diversi da quelli di ogni altra tupla*;
- `UNIQUE` garantisce che non esistano due tuple con gli *stessi valori non nulli* per gli attributi che compongono la chiave, ma i valori nulli sono ammessi (e si possono ripetere in più tuple).

Una tabella può avere più chiavi `UNIQUE`, ma una sola `PRIMARY KEY`.

La sintassi per la specifica delle chiavi ha due forme:

- in linea nella definizione di un attributo, se questo forma da solo la chiave;
- come elemento separato, dopo la specifica delle colonne (questa è l'unica opzione se la chiave è costituita da più attributi).

Osservazioni: A differenza di quanto previsto dal modello relazionale, una tabella SQL può *contenere tuple duplicate*, se non è specificata alcuna chiave. Per impedirlo, se non esiste un sottoinsieme di attributi che identifica le tuple di una relazione, può aver senso definire una chiave costituita da tutte le colonne (o, in alternativa, aggiungere un attributo codice/identificatore che abbia come unico scopo quello di fungere da chiave).

8.1 Esempi

```
CREATE TABLE Video (
  colloc DECIMAL(4) PRIMARY KEY,
  titolo VARCHAR(30) NOT NULL,
  regista VARCHAR(20) NOT NULL,
  tipo CHAR DEFAULT 'd'
);

CREATE TABLE Video (
  colloc DECIMAL(4),
  titolo VARCHAR(30) NOT NULL,
  regista VARCHAR(20) NOT NULL,
  tipo CHAR DEFAULT 'd',
  PRIMARY KEY (colloc)
);

CREATE TABLE Noleggio (
  colloc DECIMAL(4),
  dataNol DATE DEFAULT CURRENT_DATE,
  codCli DECIMAL(4) NOT NULL,
  dataRest DATE,
  PRIMARY KEY (colloc, dataNol),
  UNIQUE (colloc, dataRest)
);

CREATE TABLE Cliente (
  codCli DECIMAL(4) PRIMARY KEY,
  nome VARCHAR(20) NOT NULL,
  cognome VARCHAR(20) NOT NULL,
  telefono CHAR(15) NOT NULL,
  dataN DATE NOT NULL,
  residenza VARCHAR(30) NOT NULL,
  UNIQUE (nome, cognome, dataN)
);
```

Osservazione: Non è necessario specificare `NOT NULL` per gli attributi che compongono la chiave primaria, dato che l'obbligatorietà di tali colonne è già specificata dal vincolo di `PRIMARY KEY`.

9 Chiavi esterne

Le chiavi esterne si specificano mediante la clausola opzionale `FOREIGN KEY` del comando `CREATE TABLE`:

```
FOREIGN KEY (<lista nomi colonne>) REFERENCES <nome relazione riferita>
  [ON DELETE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]
  [ON UPDATE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]
```

Se la chiave esterna è costituita da un solo attributo, può essere specificata in linea mediante una sintassi abbreviata:

```
<specifica colonna> REFERENCES <nome relazione riferita>
  [ON DELETE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]
  [ON UPDATE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]
```

9.1 Esempi

```
CREATE TABLE Film (
  titolo VARCHAR(30),
  regista VARCHAR(20),
  anno DECIMAL(4) NOT NULL,
  genere CHAR(15) NOT NULL,
  valutaz NUMERIC(3,2),
  PRIMARY KEY (titolo, regista)
);

CREATE TABLE Video (
  colloc DECIMAL(4) PRIMARY KEY,
  titolo VARCHAR(30) NOT NULL,
  regista VARCHAR(20) NOT NULL,
  tipo CHAR NOT NULL DEFAULT 'd',
  FOREIGN KEY (titolo, regista) REFERENCES Film
);

CREATE TABLE Cliente (
  codCli DECIMAL(4) PRIMARY KEY
  -- altro...
);

CREATE TABLE Noleggio (
  colloc DECIMAL(4) REFERENCES Video,
```

```
  dataNol DATE DEFAULT CURRENT_DATE,
  codCli DECIMAL(4) NOT NULL REFERENCES Cliente
  -- altro...
);
```

9.2 ON DELETE e ON UPDATE

La clausola opzionale `ON DELETE` permette di specificare le azioni da eseguire quando vengono cancellate delle tuple *nella tabella riferita*.[2] Analogamente, la clausola opzionale `ON UPDATE` permette di specificare come gestire le modifiche dei valori della chiave primaria *nella tabella riferita*.[2]

Per entrambe le clausole, le opzioni disponibili sono:

- `NO ACTION`: la cancellazione/modifica della tupla nella tabella riferita viene eseguita solo se non ci sono riferimenti a essa nella tabella referente, altrimenti il DBMS rifiuta l'operazione e segnala un errore;
- `CASCADE`: la cancellazione/modifica della tupla nella tabella riferita implica anche la cancellazione/modifica di tutte le tuple nella tabella referente che fanno riferimento a essa.
- `SET NULL`: la tupla nella tabella riferita viene cancellata/modificata, e i valori della chiave esterna che facevano riferimento a essa vengono posti a `NULL` (se ammesso);
- `SET DEFAULT`: la tupla nella tabella riferita viene cancellata/modificata, e i valori della chiave esterna che facevano riferimento a essa vengono posti al valore di default (se specificato nella creazione della tabella).

L'opzione di default è `NO ACTION`.

Osservazione: Queste opzioni sono specificate nella creazione della tabella referente, anche se riguardano operazioni effettuate sulla tabella riferita. In questo modo, se ci sono più tabelle referenti per una stessa tabella riferita, ciascuna di esse può specificare comportamenti diversi, in base alle esigenze del dominio applicativo.

9.2.1 Esempi

```
CREATE TABLE Video (
  colloc DECIMAL(4) PRIMARY KEY,
  titolo VARCHAR(30) NOT NULL,
  regista VARCHAR(20) NOT NULL,
  tipo CHAR NOT NULL DEFAULT 'd',
```

[2] Per gli inserimenti e le modifiche *nella tabella referente* non è possibile specificare alcuna opzione: se i nuovi valori delle chiavi esterne non sono validi, l'operazione viene sempre rifiutata.

```
  FOREIGN KEY (titolo, regista) REFERENCES Film
    ON DELETE NO ACTION
);

CREATE TABLE Noleggio (
  colloc DECIMAL(4) REFERENCES Video
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  dataNol DATE DEFAULT CURRENT_DATE,
  codCli DECIMAL(4) NOT NULL REFERENCES Cliente
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  dataRest DATE,
  PRIMARY KEY (colloc, dataNol),
  UNIQUE (colloc, dataRest)
);
```

10 Cancellazione di relazioni

Il comando per la cancellazione di una relazione è:

```
DROP TABLE <nome relazione> {RESTRICT | CASCADE};
```

La clausola `RESTRICT` o `CASCADE` specifica l'azione da eseguire se ci sono altre relazioni che fanno riferimento, mediante chiavi esterne, a quella che si sta cancellando:

- con `RESTRICT`, l'operazione viene rifiutata;
- con `CASCADE`, vengono cancellate a cascata anche le tabelle referenti.

10.1 Esempio

```
CREATE TABLE Film (
  titolo VARCHAR(30),
  regista VARCHAR(20),
  anno DECIMAL(4) NOT NULL,
  genere CHAR(15) NOT NULL,
  valutaz NUMERIC(3,2),
  PRIMARY KEY (titolo, regista)
);

CREATE TABLE Video (
```

```
  colloc DECIMAL(4) PRIMARY KEY,
  titolo VARCHAR(30) NOT NULL,
  regista VARCHAR(20) NOT NULL,
  tipo CHAR NOT NULL DEFAULT 'd',
  FOREIGN KEY (titolo, regista) REFERENCES Film
);

CREATE TABLE Noleggio (
  colloc DECIMAL(4) REFERENCES Video,
  codCli DECIMAL(4) NOT NULL REFERENCES Cliente
  -- altro...
);

DROP TABLE Film RESTRICT; -- Non ha effetto
DROP TABLE Film CASCADE;  -- Cancella Film, Video e Noleggio
```

11 Modifica dello schema delle relazioni

Le modifiche allo schema di una relazione si effettuano mediante il comando

```
ALTER TABLE <nome relazione> <modifica>;
```

dove:

- `<nome relazione>` è il nome della relazione da modificare;
- `<modifica>` può essere
  - aggiunta di una nuova colonna:

    ```
    ADD [COLUMN] <specifica colonna>
    ```

  - aggiunta/modifica/rimozione del valore di default:

    ```
    ALTER [COLUMN] <nome colonna>
    {SET DEFAULT <valore default> | DROP DEFAULT}
    ```

  - eliminazione di una colonna:

    ```
    DROP [COLUMN] <nome colonna> {RESTRICT | CASCADE}
    ```

  - definizione di un vincolo (`UNIQUE`, `FOREIGN KEY`, `CHECK`, ecc.), con la possibilità di specificare un nome:

    ```
    ADD [CONSTRAINT <nome vincolo>] <specifica vincolo>
    ```

  - eliminazione di un vincolo, che deve essere dotato di nome:

```
DROP CONSTRAINT <nome vincolo> {RESTRICT | CASCADE}
```

Per le operazioni di cancellazione, la clausola `RESTRICT`~/~`CASCADE` specifica come comportarsi se ci sono eventuali elementi della base di dati (ad esempio chiavi esterne, ma non solo) che dipendono dalla colonna o dal vincolo che si vuole eliminare:

- con `RESTRICT`, che è l'opzione di default, l'operazione di eliminazione viene rifiutata;
- con `CASCADE`, anche gli elementi dipendenti vengono eliminati.

Osservazione: Eliminare un vincolo è semplice, mentre aggiungerne uno dopo la creazione della relazione è più complicato, perché eventuali dati presenti nella tabella devono essere già validi secondo il vincolo che si vuole definire.

11.1 Esempi

```
ALTER TABLE Film
ADD COLUMN studio VARCHAR(20);

ALTER TABLE Video
ALTER COLUMN tipo SET DEFAULT 'v';

ALTER TABLE Film
ADD UNIQUE (studio, titolo, anno);

ALTER TABLE Film
ADD CONSTRAINT Solo_Uno UNIQUE (studio, titolo, anno);

ALTER TABLE Film
DROP CONSTRAINT Solo_Uno;
```