Azzolini Riccardo 2019-11-12

Vincoli di integrità

1 Tipi di vincoli di integrità

Oltre ai vincoli già presentati,

- di obbligatorietà di colonne (`NOT NULL`)
- di chiave (`PRIMARY KEY` e `UNIQUE`)
- di integrità referenziale, cioè di chiave esterna (`FOREIGN KEY`)

ne esistono dei tipi più generici, legati alla semantica dei dati contenuti nelle tabelle:

- vincoli `CHECK`, **su colonna** o **su relazione**, che vengono definiti nel comando `CREATE TABLE`, ma possono anche essere aggiunti/modificati in seguito;
- **asserzioni**, che sono controlli sull'intero schema relazionale, e vengono definite come oggetti a sé stanti (ma i principali DBMS non le implementano).

Sia i vincoli `CHECK` che le asserzioni sfruttano il linguaggio di query per definire le condizioni che devono essere soddisfatte da un'istanza corretta della base di dati.

2 Vincoli `CHECK` su colonna

Accanto alla specifica di una colonna, può essere indicata, mediante la parola chiave `CHECK`, una condizione che deve essere rispettata dai valori di tale colonna, strutturata come le condizioni usate nella clausola `WHERE`. In particolare, la condizione può contenere sotto-interrogazioni, e queste possono far riferimento anche ad altre relazioni.

2.1 Esempi

```
CREATE TABLE Film (
  valutaz DECIMAL(3, 2) CHECK (valutaz BETWEEN 0.00 AND 5.00),
  -- ...
);

CREATE TABLE Video (
  tipo CHAR NOT NULL CHECK (tipo IN ('d', 'v')),
```

```
  -- ...
);
```

3 Vincoli `CHECK` su relazione

Un vincolo `CHECK` su relazione viene specificato aggiungendo, dopo la definizione delle colonne della relazione, la sintassi `CHECK (<condizione>)`. Infatti, esso può riguardare più colonne, quindi non avrebbe senso dichiararlo in seguito alla specifica di una sola di esse.

Questi vincoli vengono valutati dopo aver verificato che siano soddisfatti i vincoli sulle singole colonne.

3.1 Esempio

```
CREATE TABLE Noleggio (
  -- ...
  CHECK (dataRest >= dataNol)
);
```

4 Assegnare un nome ai vincoli

Ai vincoli (dei vari tipi: chiave, `CHECK`, ecc.) è possibile assegnare un nome, usando la sintassi `CONSTRAINT <nome>` prima della specifica del vincolo.

La specifica di un nome è utile per potersi riferire successivamente al vincolo (ad esempio, nel comando `ALTER TABLE`, al fine di modificarlo/eliminarlo).

4.1 Esempio

```
CREATE TABLE Video (
  colloc DECIMAL(4) CONSTRAINT PKey PRIMARY KEY,
  titolo VARCHAR(30) CONSTRAINT TitoloNotNull NOT NULL,
  regista VARCHAR(20) CONSTRAINT RegistaNotNull NOT NULL,
  tipo CHAR
    CONSTRAINT TipoNotNUll NOT NULL
    DEFAULT 'd'
    CONSTRAINT TipoOk CHECK (tipo IN ('d', 'v')),
  CONSTRAINT FKey FOREIGN KEY (titolo, regista)
);
```

```
ALTER TABLE Video DROP CONSTRAINT TipoOk;

ALTER TABLE Video
ADD CONSTRAINT TipoOk CHECK (tipo IN ('d', 'v', 'x'));

ALTER TABLE Video DROP CONSTRAINT RegistaNotNull;
```

5 Aggiunta e rimozione di vincoli

Quando un vincolo viene aggiunto a una tabella già esistente, mediante il comando `ALTER TABLE`, esso deve essere soddisfatto da tutti gli eventuali dati presenti nella tabella. Infatti, il nuovo vincolo viene immediatamente valutato su tutte le tuple, e la sua aggiunta viene rifiutata se esiste anche solo una tupla che non lo soddisfa.

Invece, la rimozione di un vincolo è sempre possibile.

6 Vincoli `CHECK` complessi

Mediante l'uso delle sotto-interrogazioni, un vincolo `CHECK` può esprimere condizioni arbitrarie, anche molto complesse, ma ciò:

- può rendere difficile la comprensione dello schema;
- riduce l'efficienza nella verifica dei vincoli.

Per questo, è consigliabile limitarsi a vincoli `CHECK` che fanno riferimento a singole tuple della relazione (o, comunque, a vincoli semplici).

7 Asserzioni

Per rendere più comprensibili i vincoli che si applicano a

- più tuple di una relazione
- tuple di relazioni diverse

l'SQL standard definisce le **asserzioni**. Esse sono elementi dello schema (cioè definite all'interno della base di dati, analogamente a tabelle, viste, ecc.), e vengono manipolate da appositi comandi del DDL. In particolare, la sintassi per la creazione di un'asserzione è:

```
CREATE ASSERTION <nome asserzione>
CHECK (<condizione>);
```

Mentre i vincoli `CHECK` possono semplicemente essere verificati a ogni inserimento (o modifica) effettuato nella tabella a cui sono associati, per stabilire quando verificare le asserzioni il DBMS deve individuare le tabelle coinvolte. Esse sono quindi complesse da implementare in modo efficiente. Per questo motivo, e anche perché si può spesso ottenere una funzionalità equivalente usando vincoli `CHECK` su relazioni, i principali DBMS non implementano le asserzioni.

7.1 Esempi

Uno stesso video non può essere noleggiato contemporaneamente da due clienti:

```
CREATE ASSERTION SoloUno
CHECK (NOT EXISTS (
  SELECT * FROM Noleggio
  WHERE dataRest IS NULL
  GROUP BY colloc
  HAVING COUNT(*) > 1
));
```

Un cliente non può avere più di tre video in noleggio contemporaneamente:

```
CREATE ASSERTION Max3
CHECK (NOT EXISTS (
  SELECT * FROM Noleggio
  WHERE dataRest IS NULL
  GROUP BY codCli
  HAVING COUNT(*) > 3
));
```

Un video non può essere noleggiato prima dell'uscita del film che lo contiene:

```
CREATE ASSERTION DataOk
CHECK (NOT EXISTS (
  SELECT *
  FROM Noleggio
    NATURAL JOIN Video
    NATURAL JOIN Film
  WHERE EXTRACT(YEAR FROM dataNol) < anno
));
```

8 Condizioni su valori nulli

SQL usa una logica a tre valori: `TRUE`, `FALSE` e `UNKNOWN`. `UNKNOWN` indica che il valore di verità di una condizione non è determinabile, e si ottiene come risultato della valutazione

di un *predicato semplice* (ad esempio, un operatore come `=`, `>=`, ecc.) su un attributo a valore nullo. Invece, il valore di verità di un *predicato complesso* viene calcolato in base alle seguenti tabelle di verità:

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

NOT	
TRUE	FALSE
FALSE	TRUE
UNKNOWN	UNKNOWN

Il valore `UNKNOWN` ha effetti diversi nelle interrogazioni e nei vincoli d'integrità:

- una tupla per cui la condizione di ricerca ha valore `UNKNOWN` *non viene restituita* dall'interrogazione;
- un vincolo `CHECK` la cui condizione assume valore `UNKNOWN` *non si considera violato* (perché la condizione "non è falsa").

8.1 Esempi

Nella relazione

$$\text{Noleggio}(\text{colloc}, \text{dataNol}, \text{codCli}, \text{dataRest}_o)$$

l'attributo dataRest ha valori nulli per i noleggi in corso. Di conseguenza:

- le interrogazioni

  ```
  SELECT colloc FROM Noleggio
  WHERE dataRest > CURRENT_DATE;

  SELECT colloc FROM Noleggio
  WHERE NOT dataRest < CURRENT_DATE;
  ```

  non restituiscono i noleggi in corso;

- l'interrogazione

```
SELECT colloc FROM Noleggio
WHERE dataNol > DATE '1-Nov-2018'
  OR dataRest > DATE '1-Nov-2018';
```

restituisce anche noleggi in corso, purché siano iniziati dopo il 1 novembre 2018;

- le interrogazioni

```
SELECT colloc FROM Noleggio
WHERE dataRest = CURRENT_DATE
  OR NOT dataRest = CURRENT_DATE;

SELECT colloc FROM Noleggio
WHERE dataRest = dataRest;
```

non restituiscono tutti i noleggi, ma solo quelli terminati.

9 Altre considerazioni sui valori nulli

- Nelle espressioni (ad esempio aritmetiche), se un argomento è `NULL`, allora il valore dell'intera espressione è `NULL`.
- Nel calcolo delle funzioni di gruppo, vengono escluse le tuple per cui ha valore nullo la colonna/espressione su cui la funzione è calcolata. Di conseguenza, ad esempio, `SUM(e1 + e2)` può dare un risultato diverso da `SUM(e1) + SUM(e2)`:

e1	e2	e1 + e2
1	3	4
2	NULL	NULL
NULL	NULL	NULL

`SUM(e1 + e2) = 4`
`SUM(e1) + SUM(e2) = 3 + 3 = 6`

- Una funzione di gruppo può restituire `NULL`, se applicata a un insieme vuoto (o, equivalentemente, che contiene solo valori `NULL`, dato che essi vengono rimossi prima del calcolo).
- Se `e1` ed `e2` sono `NULL`, il predicato `e1 = e2` ha valore `UNKNOWN` (e *non* `TRUE`).

10 Predicato `IS NULL`

Il predicato `IS NULL` restituisce `TRUE` se l'attributo a cui è applicato ha valore nullo, altrimenti restituisce `FALSE`. Esiste anche `IS NOT NULL`, che ha il significato opposto: `x IS NOT NULL` equivale a `NOT (x IS NULL)`.