Linguaggio SQL

Vincoli di integrità

Vincoli di integrità

- SQL permette di specificare vincoli di integrità semantica arbitrari = proprietà che i dati devono verificare

- Abbiamo già discusso la specifica di
  - Vincoli di obbligatorietà di colonne (NOT NULL)
  - Vincoli di chiave (UNIQUE e PRIMARY KEY)
  - Vincoli di integrità referenziale (chiavi esterne, FOREIGN KEY)

- Ma come possiamo ad esempio specificare che la valutazione di un film deve essere un numero compreso tra 0 e 5?

- Come possiamo specificare che non è possibile modificare la data di restituzione di un video assegnandogli una data precedente a quella memorizzata?

Vincoli di integrità

Statici

- relativi ad uno stato della base di dati
- la valutazione di un film deve essere compresa tra 0 e 5

Di transizione

- mettono in relazione stati diversi della base di dati
- non è possibile modificare la data di restituzione di un video assegnandogli una data precedente a quella memorizzata
- Nel seguito, serve il concetto di trigger

VINCOLI DI INTEGRITÀ

Vincoli statici

Su singola relazione

Su singola tupla

- Su singolo attributo
  - NOT NULL
- Su attributi multipli
  - Data_restituzione>Data_noleggio

Su tuple multiple

- vincoli di aggregazione
  - non ci sono più di 3 noleggi attivi per lo stesso cliente
- vincoli di dipendenza funzionale
  - il nome di un cliente è determinato dal suo codice

Su relazioni multiple

- un video non può essere noleggiato prima che il film che lo contiene sia uscito

Vincoli CHECK su colonna
Vincoli CHECK su relazione/tabella

ASSERZIONI in SQL
(non sempre implementati)

VINCOLI CHECK SU COLONNA

```
CREATE TABLE Film ( ...
  valutaz DECIMAL(3,2),
 ...);
```

- **Vincolo su valutazione** (singolo attributo): la valutazione è un numero compreso tra 0 e 5
  →Vincolo su singolo attributo

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

Condizione dopo CHECK = formula che potrebbe essere inserita in clausola WHERE e che si riferisce all'attributo

Può includere sotto-interrogazioni

VINCOLI CHECK SU COLONNA

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

Vincoli CHECK si possono utilizzare contemporaneamente ad altri vincoli su singolo attributo

VINCOLI CHECK SU TABELLA

- La data di restituzione di un noleggio deve seguire la data di noleggio

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

Insieme alla specifica di altli attributi

VINCOLI CHECK SU COLONNA/ TABELLA

- CREATE TABLE Suggerimento(user...,voto....,titolo...,regista...);

```
CREATE TABLE Film ( ...
   valutaz DECIMAL(3,2)
                CHECK (valutaz = (    SELECT AVG(voto)
                                      FROM Suggerimento S
                                      WHERE titolo = S.titolo
                                      AND regista = S.regista),
                ...);
```

La specifica dei vincoli può includere sottointerrogazioni che possono coinvolgere altre relazioni

E' un approccio conveniente?

VINCOLI CHECK: VERIFICHE E VIOLAZIONI

- Il vincolo viene verificato solo quando si inserisce o modifica una tupla della relazione (check su relazione) o una colonna (check su colonne)
- modifiche ad elementi diversi NON attivano la verifica
  - il vincolo può essere violato!!!!
- In caso di errore l'operazione non viene eseguita e si ha una segnalazione di errore

Esempio

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

- Ogni volta che si inserisce una nuova tupla in Film, il vincolo viene verificato
- Ogni volta che viene modificata la valutazione di un Film, il vincolo viene verificato

Esempio

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

- Ogni volta che si inserisce una nuova tupla in Noleggio, il vincolo viene verificato
- Ogni volta che viene modificata la data di restituzione o la data di noleggio, il vincolo viene verificato sulla tupla modificata

ESEMPIO

- CREATE TABLE Suggerimento(user...,voto....,titolo...,regista...);

```
CREATE TABLE Film ( ...
  valutaz DECIMAL(3,2)
            CHECK (valutaz = (   SELECT AVG(voto)
                                 FROM Suggerimento S
                                 WHERE titolo = S.titolo
                                 AND regista = S.regista    ),
                   ...);
```

- Ogni volta che si inserisce una nuova tupla in Film, il vincolo viene verificato
- Ogni volta che viene modificata la valutazione di un Film, il vincolo viene verificato
- Quando si inserisce un suggerimento (senza aggiornare valutaz del film corrispondente) non viene verificata la violazione del vincolo → il vincolo sulla valutazione non vale più

Vincoli CHECK: verifiche e violazioni - casi particolari

- Un vincolo CHECK richiede che ogni tupla nella relazione soddisfi la condizione
  - la relazione vuota soddisfa sempre tutti i vincoli CHECK
  - un vincolo CHECK non può imporre ad una relazione di contenere (almeno) un certo numero di tuple (che soddisfano una condizione)

VINCOLI CHECK: VERIFICHE E VIOLAZIONI - SUGGERIMENTI

- Usando sotto-interrogazioni è possibile ma assolutamente da evitare esprimere vincoli CHECK che verificano condizioni arbitrarie anche su altre tabelle

- I vincoli CHECK devono usare solo condizioni che si riescono a verificare esaminando la singola tupla cui associamo il vincolo
  - ⇒ migliore comprensibilità dello schema
  - ⇒ maggiore efficienza nella verifica dei vincoli

- Condizioni che richiedano di esaminare più tuple (della stessa o di altre tabelle) vanno espresse con altre modalità
  - asserzioni (si veda in seguito), che lo standard prevede ma i DBMS non supportano
  - trigger (si veda in seguito, solo per gli informatici)

Assegnare un nome ai vincoli

- Come fare per aggiungere un vincolo a una tabella esistente?
- Come fare per eliminare un vincolo da una tabella esistente?

- Operazioni possibili attribuendo un nome ai vincoli al momento della loro definizione (possibile anche per i vincoli predefiniti)

  CONSTRAINT <nomeC> CHECK (<condizione>)

- il nome dei vincoli serve a farvi riferimento per
  - modificarli
    ALTER TABLE <nomeT> ADD CONSTRAINT <nomeC> CHECK (<condizione>)
  - eliminarli
    ALTER TABLE <nomeT> DROP CONSTRAINT <nomeC>

ESEMPIO

```
CREATE TABLE  Video(
      colloc     DECIMAL(4)     CONSTRAINT PKey PRIMARY KEY,
      titolo      VARCHAR(30) CONSTRAINT Tnn NOT NULL,
      regista    VARCHAR(20) CONSTRAINT Rnn NOT NULL,
      tipo        CHAR             CONSTRAINT Snn NOT NULL
                                        DEFAULT 'd'
                                        CONSTRAINT Tok CHECK (tipo IN ('d','v'))
      CONSTRAINT FK FOREIGN KEY (titolo,regista)
                              REFERENCES Film ON DELETE RESTRICT);
```

Modifica vincolo Tok

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

Cancellazione vincolo Rnn

```
ALTER TABLE Video DROP CONSTRAINT Rnn;
```

Asserzioni

- Vincoli su tuple multiple o relazioni multiple possono essere specificati utilizzando le asserzioni
- Elementi dello schema, manipolate da appositi comandi del DDL

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

- <nome asserzione> è il nome che viene assegnato all'asserzione
- <condizione> è un predicato o una combinazione booleana di predicati (come nel caso dei vincoli CHECK)
- <condizione> in genere specifica che l'insieme delle tuple che NON soddisfano il vincolo è vuoto
  → uso del predicato NOT EXISTS

Esempio

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

= l'insieme dei video correntemente noleggiati da più di un cliente è vuoto

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

Esempio

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

  = l'insieme dei clienti che hanno più di 3 video in noleggio è vuoto

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

Esempio

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

= l'insieme dei video noleggiati prima dell'uscita del film contenuto nel video noleggiato è vuoto

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

Vincoli di integrità in postgres

- I vincoli CHECK non possono contenere sotto-interrogazioni
- Non è possibile creare asserzioni