😷

Capitolo 5 (SQL: caratteristiche evolute)

Vincoli di integrità generici

Check

Specifica di vincoli di ennupla (e anche vincoli più complessi)

`check (condizione)`

Esempio:

```
create table Impiegato (
    Matricola character(6),
    Cognome character(20),
    Nome character(20),
    Sesso character not null check (sesso in ('M', 'F')),
    Stipendio integer,
    Superiore character(6),
    check (Stipendio <= (
        select Stipendio from Impiegato J
        where Superiore = J.Matricola
    ))
)
```


Le condizioni ammissibili sono le stesse che possono apparire come argomento della clausola `where` di una interrogazione SQL

La condizione deve essere sempre verificata affinché la base di dati sia corretta

- In questo modo è possibile specificare tutti i vincoli intrarelazionali

Asserzioni

Le asserzioni rappresentano dei vincoli che non sono associati a un attributo o a una tabella in particolare, bensì appartengono direttamente allo schema

Specifica vincoli a livello di schema

`create assertion NomeAsserzione check (condizione)`

```
create assertion AlmenoUnImpiegato
    check (1 <= (select count(*) from Impiegato))
```

Mediante le asserzioni è possibile esprimere vincoli che non sarebbero altrimenti definibili

Quando un vincolo non è soddisfatto, l'operazione di modifica che ha causato la violazione viene disfatta dal sistema; questo modo di procedere è detto **rollback parziale**

Tutti i vincoli quali `not null`, `unique`, `primary key`, `foreign key`, …, descritti nel capitolo 4 sono per default verificati in modo immediato e la loro violazione causa un rollback parziale

Quando invece si rileva una violazione di un vincolo differito al termine della transazione si esegue un **rollback**, viene cioè disfatta l'intera sequenza di operazioni che costituisce la transazione

Grazie a questi meccanismi, l'esecuzione di un comando di modifica dell'istanza di una base di dati che soddisfa tutti i vincoli, immediati e differiti, produrrà sempre un'istanza della base di dati che pure soddisfa tutti i vincoli

- *Si dice anche che lo stato della base di dati è **consistente***

Viste

Le viste vengono definite in SQL associando un nome e una lista di attributi al risultato dell'esecuzione di una interrogazione

Si definisce una vista utilizzando il comando:

`create view NomeVista [(ListaAttributi)] as SelectSQL [with [local | cascaded] check option]`

```
create view ImpiegatiAmmin
    (Matricola, Nome, Cognome, Stipendio) as
    select Matricola, Nome, Cognome, Stipendio
    from Impiegato
    where Dipart = 'Amministrazione' and Stipendio > 10
```

L'interrogazione deve restituire un insieme di attributi compatibile con gli attributi nello schema della vista

L'ordine della clausola `select` deve corrispondere all'ordine degli attributi nello schema

Se gli attributi sono gli stessi non è necessario rispecificarli

Gli aggiornamenti sulle viste di solito sono ammessi solo su viste definite su una sola relazione (si incontrano problemi quando la vista è definita tramite un join tra più tabelle), alcune verifiche possono essere imposte

Non tutti i DBMS permettono la modifica diretta di una vista


```
create view ImpiegatiAmminPoveri as
    select *
    from ImpiegatiAmmin
    where Stipendio < 50
    with check option
```

- `check option` permette modifiche, ma solo a condizione che la ennupla continui ad appartenere alla vista (non posso modificare lo stipendio portandolo a 60)

Esempi di soluzioni con viste

La nidificazione nella `having` non è ammessa

```
select Dipart from Impiegato
group by Dipart
having sum(Stipendio) >= all
    (select sum(Stipendio) from Impiegato
    group by Dipart)
```

Soluzione con vista:

```
create view BudgetStipendi(Dip, TotaleStipendi) as
    select Dipart, sum(Stipendio)
    from Impiegato
    group by Dipart

select Dip
    from BudgetStipendi
    where TotaleStipendi = (select max(TotaleStipendi)
    from BudgetStipendi)
```


Altra interrogazione scorretta:

```
select avg(count(distinct Ufficio))
from Impiegato
group by Dipart
```

Soluzione con vista:

```
create view DipartUffici(NomeDip, NroUffici) as
    select Dipart, count(distinct Ufficio)
    from Impiegato
    group by Dipart;

select avg(NroUffici)
from DipartUffici
```

Controllo dell’accesso

In SQL è possibile specificare chi e come può utilizzare la base di dati (o parte di essa)

Oggetto dei **privilegi** (diritti di accesso) sono di solito le tabelle, ma anche altri tipi di risorse, quali singoli attributi, viste o domini

Un utente predefinito **_system** (amministratore della base di dati) ha tutti i privilegi

Il creatore di una risorsa ha tutti i privilegi su di essa

Privilegi

Un privilegio è caratterizzato da:

- La risorsa cui si riferisce
- L’utente che concede il privilegio
- L’utente che riceve il privilegio
- L’azione che viene permessa
- La trasmissibilità del privilegio
    - Se il privilegio può essere propagato a terzi

Tipi di privilegi offerti da SQL:

- **insert:** permette di inserire nuovi oggetti (ennuple)
- **update:** permette di modificare il contenuto
- **delete:** permette di eliminare oggetti
- **select:** permette di leggere la risorsa
- **references:** permette la definizione di vincoli di integrità referenziale verso la risorsa (può limitare la possibilità di modificare la risorsa)
- **usage:** permette l'utilizzo in una definizione (per esempio, di un dominio)

Grant e revoke

Concessione di privilegi:

```
grant < privileges | all privileges > on Resource to User [with grant option]
```

- `grant option` specifica se il privilegio può essere trasmesso ad altri utenti
  - `grant select on Diparimento to Stefano`

Revoca di privilegi:


```
revoke privileges on Resource from Users [restrict | cascade]
```

- `cascade` rimuove i privilegi anche a coloro cui sono stati propagati

Transazioni (libro)

Una **transazione** identifica una unità elementare di lavoro svolta da una applicazione, cui si vogliono associare particolari caratteristiche di correttezza, robustezza e isolamento

In particolari sono utili con riferimento a operazioni che modificano il contenuto della base di dati

Un sistema che mette a disposizione un meccanismo per la definizione e l'esecuzione di transazioni con le caratteristiche suddette viene detto **sistema transazionale**

Una transazione può essere definita sintatticamente:

- Ogni transazione è specificata racchiudendo la sequenza di operazioni che la compongono all'interno di una coppia di istruzione che ne specificano l'inizio e la conclusione

Esempio di transazione in SQL che trasferisce 10 unità da un conto corrente:

```
start transaction;
update ContoCorrente
    set Saldo = Saldo + 10
    where NumConto = 12202;
update ContoCorrente
    set Saldo = Saldo - 10
    where NumConto = 42177;
commit work;
```

L'istruzione `commit work` specifica il fatto che si richiede una conclusione positiva della transazione e che tutti gli aggiornamenti devono essere salvati nella base di dati

In alcune applicazioni la sequenza di operazioni potrebbe essere annullata

- Allo scopo è disponibile un'altra istruzione, la `rollback work` (o semplicemente `rollback`)

Nell'esempio precedente si potrebbe fare tale scelta se il saldo del conto corrente da cui si preleva risultasse negativo:

```
start transaction;
update ContoCorrente
    set Saldo = Saldo + 10
    where NumConto = 12202;
update ContoCorrente
    set Saldo = Saldo - 10
```

```
    where NumConto = 42177;
select Saldo into A
    from ContoCorrente
    where NumConto = 42177;
if A >= 0
    then commit work;
    else rollback work;
commit work;
```

Transazione

Insieme di operazioni da considerare indivisibile (”atomico”), corretto anche in presenza di concorrenza e con effetti definitivi

Presente le seguenti proprietà (”acide”) - Il termine è un acronimo derivante dall’inglese, ove ACID denota le iniziali di “Atomicity, Consistency, Isolation, Durability”:

- **A**tomicità
- **C**onsistenza (correttezza anche in presenza di concorrenza)
- **I**solamento
- **D**urabilità (persistenza)

Atomicità:

- La sequenza di operazioni sulla base di dati viene eseguita per intero o per niente:
- Trasferimento di fondi da un conto A ad un conto B: o si fanno il prelevamento da A e il versamento su B o nessuno dei due

Consistenza:

- Al termine dell’esecuzione di una transazione, i vincoli di integrità debbono essere soddisfatti

- “Durante” l’esecuzione ci possono essere violazioni, ma se restano alla fine allora la transazione deve essere annullata per intero (”abortita”)

Isolamento:

- L’effetto di transazioni concorrenti deve essere coerente (ad esempio “equivalente” all’esecuzione separata)
  - Se due assegni emessi sullo stesso conto corrente vengono incassati contemporaneamente si deve evitare di trascurarne uno

Durabilità:

- La conclusione positiva di una transazione corrisponde ad un impegno (in inglese **commit**) a mantenere traccia del risultato in modo definitivo, anche in presenza di guasti e di esecuzione concorrente

Transazioni in SQL

Istruzioni fondamentali:

- `begin transaction`: specifica l’inizio della transazione (le operazioni non vengono eseguite sulla base di dati)
- `commit work`: le operazioni specificate a partire dal `begin transaction` vengono eseguite
- `rollback work`: si rinuncia all’esecuzione delle operazioni specificate dopo l’ultimo `begin transaction`

Esempio:

```
begin transaction;
update ContoCorrente
    set Saldo = Saldo - 10
    where NumeroConto = 12345:
update ContoCorrente
    set Saldo = Saldo + 10
```

```
    where NumeroConto = 55555;
commit work;
```

Verranno eseguite entrambe o nessuna delle due