Capitolo 5

Esercizio 5.1

Definire sulla tabella `Impiegato` il vincolo che il dipartimento `Amministrazione` abbia meno di 100 dipendenti, con uno stipendio medio superiore ai 40 mila €.

Soluzione:

```
check (100 >= ( select count(*)
          from Impiegato
                 where Dipartimento='Amministrazione' )
       and 40000 <= ( select avg(Stipendio)
                      from Impiegato
                      where       Dipartimento='Amministrazione'))
```

Esercizio 5.2 Definire (con una opportuna notazione) su una relazione PAGHE (Matricola, StipLordo, Ritenute, StipNetto, OK) un vincolo che imponga che il valore di OK è:

- zero se StipNetto è pari alla differenza fra StipLordo e Ritenute
- uno altrimenti.

Soluzione

```
    (Verifica = 0 and (Netto = StipLordo-Tasse))
or
    (Verifica = 1 and (Netto <> StipLordo-Tasse)).
```

Esercizio 5.3

Definire a livello di schema il vincolo che il massimo degli stipendi degli impiegati di dipartimenti con sede a Firenze sia minore dello stipendio di tutti gli impiegati del dipartimento Direzione.

Soluzione:

```
create assertion ControlloSalari
  check ( not exists(    select *
              from Impiegato join Dipartimento on
              Impiegato.Dipartimento=Dipartimento.Nome
              where Dipartimento.Città='Firenze' and
              Stipendio > (  select min(Stipendio)
                        from Impiegato
                        where
     Dipartimento='Direzione')
         )        )
```

Esercizio 5.4 Indicare quali delle seguenti affermazioni sono vere.

1. Nei sistemi relazionali le viste possono essere utili al fine di rendere più semplice la scrittura delle interrogazioni.
2. Nei sistemi relazionali le viste possono essere utili al fine di rendere più efficienti le interrogazioni.
3. Nei sistemi relazionali le viste introducono ridondanze.

Soluzione

1. Nei sistemi relazionali le viste possono essere utili al fine di rendere più semplice la scrittura delle interrogazioni. **VERO**
2. Nei sistemi relazionali le viste possono essere utili al fine di rendere più efficienti le interrogazioni. **VERO**
3. Nei sistemi relazionali le viste introducono ridondanze. **FALSO**

Esercizio 5.5

Dato il seguente schema:

```
AEROPORTO(Città, Nazione,NumPiste)
VOLO(IdVolo,GiornoSett,CittàPart,OraPart,
        CittàArr,OraArr,TipoAereo)
AEREO(TipoAereo,NumPasseggeri,QtaMerci)
```

scrivere, facendo uso di una vista, l'interrogazione SQL che permette di determinare il massimo numero di passeggeri che possono arrivare in un aeroporto italiano dalla Francia di giovedì (se vi sono più voli, si devono sommare i passeggeri).

Soluzione:

```
create view Passeggeri(Numero)
  as select sum ( NumPasseggeri )
  from AEROPORTO as A1 join VOLO on A1.Città=CittàPart
     join AEROPORTO as A2 on A2.Città=CittàArr
  capitolo5_viste       join             AEREO             on
  VOLO.TipoAereo=Aereo.TipoAereo
      where A1.Nazione='Francia' and A2.Nazione='Italia'
          and GiornoSett='Giovedì'
      group by A2.Città

  select max(Numero)
  from Passeggeri
```

Esercizio 5.6

Definire una vista che mostra per ogni dipartimento il valore medio degli stipendi superiori alla media del dipartimento

Soluzione:

```
create view SalariSopraMedia (Dipartimento,Stipendio) as
select Dipartimento, avg(Stipendio)
from I
where Stipendio > ( select avg(Stipendio)
                    from Impiegato as I )
                where I.Dipartimento=I.Dipartimento )
group by Dipartimento
```

Esercizio 5.7 Dato il seguente schema relazionale:

- DIPENDENTE (CodiceFiscale, Cognome, Nome)
- PROFESSORE (CodiceFiscale, Qualifica, Anzianità,Facoltà*) con vincolo di integrità referenziale fra l'attributo CodiceFiscale e la relazione DIPENDENTE e fra l'attributo Facoltà e la relazione FACOLTÀ
- FACOLTÀ (Codice, Nome, Indirizzo)
- CORSODISTUDIO (Codice, Nome, Facoltà, Presidente) con vincolo di integrità referenziale fra l'attributo Facoltà e la relazione FACOLTÀ a fra l'attributo Presidente e la relazione PROFESSORE
- COLLABORAZIONE (CorsoDiStudio, Facoltà, Professore, Tipo) con vincolo di integrità referenziale fra l'attributo CorsodiStudio, Facoltà e la relazione CORSODISTUDIO e fra l'attributoProfessore e la relazione PROFESSORE
- CORSO (Codice, Materia, Docente, Semestre) con vincolo di integrità referenziale fra l'attributo Materia e la relazione MATERIA e fra Docente e la relazione PROFESSORE
- MATERIA (Sigla, Nome)

formulare le interrogazioni in SQL:

1. mostrare i professori, con codice fiscale, cognome, cognome, qualifica, anzianità e nome della eventuale facoltà di afferenza (per i professori che non afferiscono ad alcuna facoltà dovrà comparire il valore nullo)
2. trovare cognome e qualifica dei professori che afferiscono alla stessa facoltà di un professore chiamato Mario Bruni di qualifica "ordinario"
3. trovare i codici delle facoltà cui non afferisce alcun professore con cognome Bruni e qualifica "ordinario".

Soluzione
Poiché tutte le interrogazioni richiedono (anche più volte), il join di DIPENDENTE e PROFESSORE, è utile la vista:

```
create view prof as
    select d.cf, cognome, nome, qualifica,
           anzianita, facolta
    from dipendente d join professore p on d.cf=p.cf.
```

Interrogazioni:

1. ```
   select cf, cognome, qualifica,
        anzianita facolta.nome as facolta
   from prof left join facolta
                  on facolta=codice;
   ```

2. ```
   select distinct p1.cognome, p1.qualifica
   from prof p1 join prof p2
               on p1.facolta=p2.facolta
   where p2.cognome='Bruni'
   and p2.nome='Mario'
   ```

```
and p2.qualifica='Ordinario';
```

3.
```
select codice as codicefacolta
from facolta
where codice not in (
        select facolta
        from  prof
        where cognome='Bruni'
        and qualifica='Ordinario').
```

Esercizio 5.8 Considerare la base di dati relazionale definita per mezzo delle seguenti istruzioni (è lo schema già visto nell'esercizio 4.16):

```
create table Studenti (
       matricola numeric not null primary key,
       cognome char(20) not null,
       nome char(20) not null,
       eta numeric not null
);
create table Esami (
       codiceCorso numeric not null,
       studente numeric not null
           references Studenti(matricola),
       data date not null,
       voto numeric not null,
       primary key (codiceCorso, studente, data)
).
```

Formulare in SQL l'interrogazione che trova lo studente con la media più alta.
Soluzione

```
create view StudMedia as
        select studente, avg(voto) as media
        from Esami
        group by Studente;

select studente
from StudMedia
where media >= all (
    select media
    from StudMedia ).
```

Esercizio 5.9 Considerare la seguente base di dati relazionale:

- VENDITE (NumeroScontrino, Data)
- CLIENTI (Codice, Cognome, Età)
- DETTAGLIVENDITE (NumeroScontrino, Riga, Prodotto, Importo, Cliente) con valori nulli ammessi sull'attributo Cliente e con vincoli di integrità referenziale fra NumeroScontrino e la relazione VENDITE e fra Cliente e la relazione CLIENTI;

formulare in SQL:

1. l'interrogazione che restituisce i prodotti acquistati in ciascuna data (che mostra cioè le coppie $< p, d >$ tali che il prodotto p è stato acquistato nella data d;
2. l'interrogazione che restituisce i prodotti che sono stati acquistati in due date diverse;
3. la vista VENDITECONTOTALE(NumeroScontrino, Totale), che riporta, per ogni scontrino l'importo totale (ottenuto come somma degli importi dei prodotti riportati sullo scontrino).

Soluzione
Definiamo innanzitutto una vista:

1.
```
create view VD(P, D) as
select Prodotto, Data
from Vendite V join DettagliVendite D
             on V.Numeroscontrino = D.NumeroScontrino;

select distinct P, D
from VD;
```

2.
```
select P
from VD as VD1
where not exists (
    select *
    from VD as VD2
    where VD1.P=VD2.P and VD1.D<>VD2.D);
```

3.
```
create view VenditeConTotale as
     select NumeroScontrino, sum(Importo) As Totale
     from DettagliVendite
     group by NumeroScontrino.
```

Esercizio 5.10 Considerare la seguente base di dati relazionale:

- PERSONE (FC, Cognome, Nome, Età)
- IMMOBILI (Codice, Via, NumeroCivico, Città, Valore)
- PROPRIETÀ (Persona, Immobile, Percentuale) con vincolo di di integrità referenziale fra l'attributo Immobile e la relazione PERSONE e fra l'attributo Immobile e la relazione IMMOBILI.

Nota: l'attributo Percentuale indica la percentuale di proprietà.
Definire in SQL:

- la vista definita per mezzo della seguente espressione dell'algebra relazionale: Vista = Immobili $\bowtie_{Codice=Immobile}$ Proprietà
- l'interrogazione che fornisce codici fiscali, nome e cognome delle persone che posseggono un solo immobile e lo posseggono a 100%
- l'interrogazione che fornisce, per ciascuna persona, il codice fiscale, il nome, il cognome e il valore complessivo degli immobili di sua proprietà (dove il valore è la somma dei valori ciascuno pesato con la percentuale di proprietà: se Tizio possiede un immobile di valore 150 al 100% e uno di valore 200 al 50%, allora il valore complessivo sarà (150 x 100)/100 + (200 x 50)/100 = 250).

Soluzione

1. ```
   create view ProprImmobili(Codice, Via, NumeroCivico,
                                 Citta, Valore, Persona,
                                 Percentuale) as
       select Codice, Via, NumeroCivico, Citta, Valore,
              Persona, Percentuale
       from Immobili, Proprieta
       where Codice = Immobile);
   ```
2. ```
   select CF, Cognome, Nome
   from ProprImmobili, Persone
   where Persona = CF
   and Percentuale = "100%"
   and Codice <> all (
       select Proprietario
       from ProprImmobili PI1, ProprImmobili PR2
       where PI1.Codice <> PI2.Codice
       and PI1.Proprietario = PI2.Proprietario)
   ```
3. ```
   select CF, Cognome, Nome, sum(Valore * Percentuale / 100)
   from Persone join Proprieta
           on (CF = Persona)
               join Immobili
                   on (Codice = Immobile
   group by (CF, Cognome, Nome).
   ```
   ```

Esercizio 5.11

Si supponga di avere le tabelle:

Magazzino(Prodotto, QtaDisp, Soglia, QtaRiordino)
OrdineInCorso(Prodotto, Qta)

Scrivere una procedura SQL che realizza il prelievo dal magazzino accettando 2 parametri, il prodotto `prod` e la quantità da prelevare `QtaPrelievo`. La Procedura deve verificare inizialmente che `QtaPrelievo` sia inferiore al valore di `QtaDisp` per il prodotto indicato. `QtaPrelievo` viene quindi sottratta al valore di `QtaDisp`. A questo punto la procedura verifica se per il prodotto `QtaDisp` risulta minore di `Soglia`, senza che in `OrdineInCorso` compaia già una tupla relativa al prodotto prelevato; se si, viene inserito un nuovo elemento nella tabella `OrdineInCorso`. Con i valori di `Prod` e del corrispondente attributo `QtaRiordino`.

Soluzione:

```
#include<stdlib.h>
main()
{
exec sql begin declare section;
     char Prodotto, prod;
     int QtaDisp, Soglia, QtaRiordino, Qta, Qta1, i;
exec sql end declare section;

exec sql declare MagazzinoCursore cursor for
     select  Prodotto, QtaDisp, Soglia, QtaRiordino
     from Magazzino;
exec sql open MagazzinoCursore;

prod = sceltaProdotto();
Qta= sceltaQta();
i=0;

do{
exec sql fetch MagazzinoCursore into
     :Prodotto, :QtaDisp, :Soglia, :QtaRiordino;
if (Prodotto == prod) i=1;
}while(Prodotto == prod || sqlca.sqlcode==0 )

if (i=1){
     printf("ERRORE – Prodotto non trovato");
     exit(1);
     }
if (QtaDisp < Qta){
     printf("ERRORE - Quantità non disponibile");
     exit(1);
     }
QtaDisp = QtaDisp – Qta;

if (QtaDisp < Soglia){
     Qtariordino = QtaRiordino + QtaDisp - Soglia;
     }


exec sql update magazzino
```

```
     set QtaDisp = :QtaDisp
     set QtaRiordino = :QtaRiordino
     where current of MagazzinoCursore;

exec sql declare OrdineCursore cursor for
     select  Prodotto, Qta
     from OrdineInCorso;
exec sql open OrdineCursore ;

i=0;
do{
exec sql fetch  OrdineCursore into
     :Prodotto, :Qta1;
if (Prodotto == prod) i=1;
}while(Prodotto == prod || sqlca.sqlcode==0 )

if (i=1){
     exec sql update OrdineInCorso
          set Qta = :QtaRiordino;
          where current of OrdineCursore;
     }
else exec sql insert into OrdineInCorso
     values(:prod,:QtaRiordino)";

}
```

Esercizio 5.12

Dato lo schema relazionale:

IMPIEGATO (Nome, Salario, DipNum)
DIPARTIMENTO (DipNum, NomeManager)

Definire le seguenti regole attive in Oracle e DB2:

1. una regola, che quando il dipartimento è cancellato, mette ad un valore di default (99) il valore di DipNum degli impiegati appartenenti a quel dipartimento;
2. una regola che cancella tutti gli impiegati appartenenti a un dipartimento quando quest'ultimo è cancellato;
3. una regola che, ogni qual volta il salario di un impiegato supera il salario del suo manager, pone tale salario uguale al salario del manager;
4. una regola che, ogni qual volta vengono modificati i salari, verifica che non vi siano dipartimenti in cui il salario medio cresce più del tre per cento, e in tal caso annulla la modifica.

Soluzione:

I 4 trigger hanno la stessa sintassi sia per Oracle che per DB2

1)
```
create trigger T1
after delete on DIPARTIMENTO
for each row
when (exists (select *
              from IMPIEGATO
              where DipNum=Old.DipNum))
udpade IMPIEGATO.DipNum = 99
```

2)
```
create trigger T2
after delete on DIPARTIMENTO
for each row
when (exist (select *
             from IMPIEGATO
             where DipNum=Old.DipNum))
delete from IMPIEGATO where DipNum=Old.DipNum
```

3)
```
create trigger T3
after update of Salario on IMPIEGATO
for each row
     declare x number;
     begin
          select Salary into x
          from IMPIEGATO join DIPARTIMENTO on
                         Nome = NomeManager
          Where DIPARTIMENTO.DipNum = New.DipNum
          if new.Salario > x then
               update IMPIEGATO set Salario = x
               where Nome = New.Nome
     end
```

4)
```
create trigger T4
after update of Salario on IMPIEGATO
for each row
     declare x number;
     declare y number;
     declare l number;
     begin
          select avg(salario), count(*) into x,l
          from IMPIEGATO
          where DipNum=new.DipNum;
          y=((x*l)-new.Salario+old.Salario)/l;
          if (x>(y*1.03)) then
          update IMPIEGATO set Salario=old.Salario
          where DipNum=new.DipNum;
     end
```

Esercizio 5.13

Riferendosi alla base di dati dell'esercizio precedente, definire in DB2 e in Oracle un trigger R_1 che, quando è cancellato un impiegato che svolge il ruolo di manager di un dipartimento, cancella quel dipartimento e tutti i suoi dipendenti. Definire inoltre un trigger R_2 che, ogni qual volta vengono modificati i salari, verifica la loro media, e se essa supera i 50.000 cancella tutti gli impiegati il cui salario è stato modificato e attualmente supera gli 80.000. Si consideri poi uno stato di base di dati con sei impiegati: Giovanna, Maria, Andrea, Giuseppe, Sandro e Carla, in cui:

- Giovanna è manager del dipartimento 1, in cui lavorano Giovanna, Maria e Giuseppe;
- Maria è Manager del dipartimento 2, in cui lavora Andrea.
- Giuseppe è manager del dipartimento 3, in cui lavorano Sandro e Carla.

Si assuma infine una transazione che cancella l'impiegata Giovanna e modifica i salari in modo tale che la loro media ecceda i 50.000 e il salario di Maria dopo le modifiche ecceda 80.000. Descrivere l'operato dei trigger.

Soluzione:

La soluzione per il trigger R_1 è uguale sia per Oracle che per DB2.

```
create trigger R1
after delete on IMPIEGATO
for each row
when (Old.No in( select NomeManager
                 from DIPARTIMENTO )
begin
     delete from IMPIEGATO where DipNum = Old.DipNum;
     delete from DIPARTIMENTO where DipNum = Old.DipNum;
end
```

La soluzione per il trigger R_2 è diversa per i due DBMS.

```
(ORACLE)
create trigger R2Oracle
after update of Salario on IMPIEGATO
for each statement
when ( (select avg(Salario) from new_table) > 50000)
delete from IMPIEGATO
where Salario > 80000
and Nome in ( Select new_table.Nome
              from new_tableas n join old_table as o
                   on n.Nome = o.Nome
              where n.Salario <> o.Salario)

(DB2)
create trigger R2DB2
after update of Salario on IMPIEGATO
when ( (select avg(Salario) from new_table) > 50000)
delete from IMPIEGATO
where Salario > 80000
and Nome in ( Select new_table.Nome
```

```
from new_tableas n join old_table as o
     on n.Nome = o.Nome
where n.Salario <> o.Salario)
```

Assegnando il seguente stato iniziale alla base di dati:

IMPIEGATO

Nome	Salario	DipNum
Giovanna	60000	1
Maria	50000	1
Giuseppe	50000	1
Andrea	40000	2
Sandro	40000	3
Carla	40000	3

DIPARTIMENTO

DipNum	NomeManager
1	Giovanna
2	Maria
3	Giuseppe

Transazione SQL:

```
delete from IMPIEGATO where Nome = ”Giovanna”
```
update IMPIEGATO set Salario = Salario * 1,2
update IMPIEGATO set Salario = 85000 where Nome = “Maria”

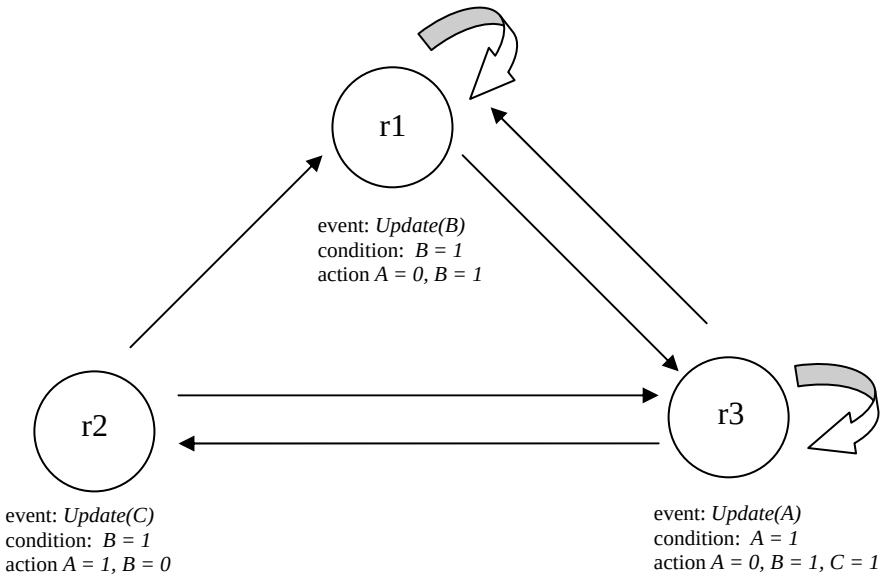
Il comportamento della base dati dell’esempio è molto semplice. Il trigger R1 reagisce alla cancellazione dell’impiegato Giovanna cancellando i dipartimento e tutti i suoi impiegati. Gli impiegati di Giovanna erano Maria e Giuseppe. La cancellazione di questi dipendenti causa nuovamente l’esecuzione del trigger R1, che cancella gli impiegati Andrea, Sandro e Carla e i loro Dipartimenti. Come si può vedere il trigger degenera e cancella tutti gli impiegati della base dati.
Le successive operazioni non modificano il DB.

Esercizio 5.14

Mostrare il grafo di attivazione delle seguenti regole, descritte in forma sintetica, e discutere quindi la loro terminazione:

- r1:
  - event: *Update(B)*
  - condition: *B = 1*
  - action *A = 0, B = 1*
- r2:
  - event: *Update(C)*
  - condition: *B = 1*
  - action *A = 1, B = 0*
- r3:
  - event: *Update(A)*
  - condition: *A = 1*
  - action *A = 0, B = 1, C = 1*

Soluzione



Analizziamo ogni singola regola attiva del database.
La regola r1 si attiva quando avvengono delle modifiche su B e se B=1 esegue le modifiche descritte. Queste modifiche fanno sospendere il trigger e partire il nuovo trigger sulla regola r3, che gestisce la risorsa A. Questa non fa nulla in quanto la condizione A = 1 non viene verificata. L'esecuzione torna a r1 che setta B = 1 ed entra in ciclo infinito.
La regola r2 si verifica quando si ha un aggiornamento della risorsa C. La condizione è che B sia uguale a 1. Se non è verificata il trigger non fa nulla. Se la condizione si verifica il trigger esegue

l’azione ed esegue la regola r3 che controlla la risorsa A. Se A = 1 la regola setta la risorsa A a 0, imposta B a 1, facendo partire il trigger sulla regola r1 che mette A a 0 (come già era) e torna a r3 che imposta C = 1 facendo rieseguire la regola r2 ed entra in un ciclo infinito.
La regola r3 si attiva sugli eventi sulla risorta A e cicla con la regola r2 come nell’esempio precedente.

Esercizio 5.15

Dato lo schema relazionale:

DOTTORANDO (Nome, Disciplina, Relatore)
PROFESSORE (Nome, Disciplina)
CORSO (Titolo, Professore)
ESAMI (NomeStud, TitoloCorso)

Descrivere i trigger che gestiscono i seguenti vincoli di integrità (*business rules*):

1. ogni dottorando deve lavorare nella stessa area del suo relatore;
2. ogni dottorando deve aver sostenuto almeno 3 corsi nell'area del suo relatore;
3. ogni dottorando deve aver sostenuto l'esame del corso di cui è responsabile il suo relatore.

Soluzione:

1)
```
create trigger T1
after update of Disciplina on DOTTORANDO
for each row
when Disciplina <> select Disciplina
     from PROFESSORE
     where PROFESSORE.Nome=new.Relatore
then signal SQLSTATE 70005 ( "Disciplina sbagliata" )
```

2)
```
create trigger T2
after update of Disciplina on DOTTORANDO
for each row
when 3 < ( select count(*)
     from ESAMI join CORSO on TitoloCorso = Titolo
          join PROFESSORE on Professore = PROFESSORE.Nome
          join DOTTORANDO on NomeStud = DOTTORANDO.Nome
          join PROFESSORE as P2 on Relatore = P2.Nome
     where PROFESSORE.Disciplina = P2.Disciplina
          and DOTTORANDO.Nome=new.Nome )
then signal SQLSTATE 70006 ("Almeno 3 corsi")
```

3)
```
create trigger T3
after update of Relatore on DOTTORANDO
for each row
when not exists ( select *
                  from ESAME join CORSO on TitoloCorso = Titolo
                  where NomeStud = new.Nome and
                      Professore = new.Relatore )
then signal SQLSTATE 70007 ("Esame mancante")
```

Esercizio 5.16

Tramite la definizione di una vista permettere all'utente "Carlo" di accedere al contenuto di Impiegato, escludendo l'attributo Stipendio.

Soluzione:

Ipotizzando la tabella Impiegato

```
   Impiegato(Codice, Nome, Cognome, Stipendio, Dipartimento)
create view ImpiegatoRistretto
(Codice,Nome,Cognome,Dipartimento) as
select  Codice,Nome,Cognome,Dipartimento
from Impiegato

grant select on ImpiegatoRistretto to Carlo
```

Esercizio 5.17

Descrivere l'effetto delle seguenti istruzioni: quali autorizzazioni sono presenti dopo ciascuna istruzione? (ciascuna linea è preceduta dal nome dall'utente che esegue il comando)

```
Stefano:  grant select on Table1 to Paolo,Riccardo
               with grant option
Paolo:         grant select on Table1 to Piero
Riccardo: grant select on Table1 to Piero with grant option
Stefano:       revoke select on Table1 from Paolo cascade
Piero:         grant select on Table1 to Paolo
Stefano:       revoke select on Table1 from Riccardo cascade
```

Soluzione:

1. Stefano concede a Paolo e a Riccardo l'autorizzazione di `select` e di concedere a loro volta l'autorizzazione
2. Paolo concede a Piero l'autorizzazione di `select`
3. Riccardo concede a Piero l'autorizzazione di `select` e di `grant`. Ora Piero ha 2 diverse autorizzazioni sulla tabella.
4. Stefano revoca l'autorizzazione data a Paolo. A causa dell'attributo `cascade` anche Piero perde le autorizzazioni concesse da Paolo ma continua ad avere quella concessa da Riccardo.
5. Ora Paolo può di nuovo accedere alla tabella grazie all'autorizzazione concessa da Piero
6. Stefano revoca l'autorizzazione di Riccardo e tramite `cascade` anche di Piero e di Paolo. Ora solo Stefano ha autorizzazioni sulla tabella.

Esercizio 5.18 Considerare i seguenti vincoli di integrità:

- a

```
CHECK ((Crediti = 0 AND Voto < 18) OR
(Crediti > 0 AND Voto >= 18))
```

- b

```
CHECK (Crediti > 0 AND Voto >= 18)
```

- c

```
CHECK (Crediti = 0 AND Voto < 18)
```

- d nessuna delle precedenti

e le seguenti specifiche:

1. sono registrati solo gli esami superati (con voto pari almeno a 18) e i crediti sono sempre positivi
2. il voto è pari almeno a 18 se e solo se i crediti sono maggiori di zero
3. se il voto è pari almeno a 18 i crediti sono positivi, se il voto è inferiore a 18 non c'è vincolo sui crediti
4. nessuna delle precedenti.

Abbinare vincoli e specifiche.
Soluzione

1. b
2. a
3. d
4. c

Esercizio 5.19 Indicare quali tra le seguenti affermazioni sono vere:

1. tra viste è possibile definire vincoli di integrità referenziale
2. possibile inserire record in viste utilizzando operazioni DDL
3. le possono essere ottenute esclusivamente come risultato di una valutazione di una query
4. l'utilizzo di viste può consentire il miglioramento delle prestazioni nell'esecuzione di una query
5. non è possibile definire una vista con record duplicati
6. la modifica dei dati a cui si riferisce la query che genera una vista implica la modifica del risultato della vista stessa.

Soluzione

1. Falso
2. Falso
3. Vero
4. Vero
5. Falso
6. Falso

Esercizio 5.20 Indicare quali tra le seguenti affermazioni sono vere:

1. sulle viste è possibile definire vincoli di dominio
2. le viste possono essere utilizzate per semplificare sintatticamente la scrittura di una query
3. la DDL "INSERT INTO nomeVista" permette l'inserimento di un record in una vista
4. la cancellazione di una vista implica la cancellazione nelle tabelle originarie di tutti i dati riportati
5. è necessario definire a priori la chiave primaria di una vista
6. l'esecuzione di una query su viste è sempre più efficiente dell'esecuzione della stessa query su tabelle poiché le viste non sono materializzate.

Soluzione

1. Falso
2. Vero
3. Falso
4. Falso
5. Falso
6. Falso

Esercizio 5.21 Nell'assunzione che queste due query siano sintatticamente e semanticamente corrette, dedurre lo schema della vista STUDENTI e della tabella PERSONA.

```
create view Studenti as
   select Nome, Cognome, 'studente', Eta
   from Persona
   where
       Mansione <> 'dipendente';

select *
from Studenti
  minus
select *
from Persone
where Mansione = 'Studente'.
```

Quale relazione esiste fra la cardinalità della vista STUDENTI e la cardinalità della relazione PERSONE?
Soluzione

- STUDENTI (Nome, Cognome, Mansione, Età)
- PERSONE (Nome, Cognome, Mansione, Età)

La cardinalità delle due relazioni risulta essere:
$|\text{PERSONE}| >= |\text{STUDENTI}|$;
vale l'uguale nell'ipotesi di assenza di dipendenti.

Esercizio 5.22 Sia dato il seguente schema relazionale:

- FILM (Titolo, Anno, Genere)
- ATTORE (Cognome, Nome, Nazionalità)
- PARTECIPAZIONE (CognomeAttore, NomeAttore, TitoloFilm, Compenso) con vincoli di integrità referenziale fra gli attributi CognomeAttore, NomeAttore e la relazione ATTORI e fra l'attributo TitoloFilm e la relazione FILM.

Definire in SQL, anche attraverso l'uso di viste:

1. l'interrogazione che trova gli attori che hanno partecipato ad almeno un film:
2. l'interrogazione che trova gli attori che hanno partecipato ad almeno cinque film;
3. l'interrogazione che trova i cognomi di tutti gli attori che hanno partecipato ad un film insieme a Sylvester Stallone.

Soluzione

1.
```
select Cognome, Nome
from Attore a
where exists (
    select *
    from Partecipazione
    where a.Cognome = NognomeAttore
    and a.Nome = NomeAttore);
```

2.
```
create view AttoriFilm as
    select Cognome, Nome, count(*) as NumeroFilm
    from Attore a join Partecipazione p
             on (a.Cognome = p.CongnomeAttore
                 and a.Nome = p.NomeAttore)
    group by Cognome, Nome;

select Cognome, Nome
from AttoriFilm
where NumeroFilm >=5;
```

3.
```
create view FilmStallone as
    select Titolo
    from Partecipazione
    where CognomeAttore = 'Stallone'
    and NomeAttore = 'Sylvester';

select Cognome
from Attori join Partecipazione
          on (Cognome = CognomeAttore
              and Nome = NomeAttore)
```

```
where Titolo not in (
       select Titolo
       from FilmStallone).
```

Esercizio 5.23 Dato il modello relazionale seguente:

- SPEDIZIONE (Pacco, Mittente, Destinatario, DataStimata, DataEffettiva)
- PACCO (IdPacco, TipoMerceologico, GradoFragilità, Peso)
- TIPOMERCEOLOGICO (IdTipoMerceologico, Descrizione, PolizzaAssicurativa)
- ASSICURAZIONE (IdAssicurazione, Nome, Indirizzo, Coefficiente)
- UTENTE (IdUtente, Cognome, Nome, Indirizzo)

scrivere in SQL:

1. l'interrogazione che trova tutti i pacchi spediti a Paolo Rossi
2. l'interrogazione che trova tutti i pacchi spediti da Paolo Rossi a Mario Bruni
3. l'interrogazione che trova i cognomi di tutti gli utenti che hanno spedito almeno due pacchi con la assicurazione SECUR.

Soluzione

1.
```
select Pacco.*
from Pacco, Spedizione, Utente
where Pacco.Destinatario = Utente.IdUtente
and Utente.Cognome = 'Rossi'
and Utente.Nome = 'Paolo';
```

2.
```
select Pacco.*
from Pacco, Spedizione, Utente u1, Utente u2
where Pacco.Destinatario = u1.IdUtente
and Pacco.Mittente = u2.IdUtente
and u2.Cognome = 'Rossi'
and u2.Nome = 'Paolo'
and u1.Cognome = 'Bruni'
and u1.Cognome = 'Mario';
```

3.
```
create view Pacchi_spediti_con_SECUR as
     select idUtente, count (*) as NumeroPacchi
     from Spedizione S, Pacco P, TipoMerceologico TM,
          Assicurazione A, Utente U
     where P.Mittente = U.IdUtente
     and   S.Pacco = P.IdPacco
     and   TM.IdTipoMerceologico = P.TipoMerceologico
     and   A.IdAssicurazione = TM.PolizzaAssicurativa
     and   Assicurazione.Nome = 'SECUR'
     group by IdUtente;

select Utente.Cognome
from Pacchi_spediti_con_SECUR v, Utente
where NumeroPacchi >= 2
and Utente.IdUtente = v.IdUtente.
```

Esercizio 5.24 Dato lo schema relazionale dell'esercizio 5.18 scrivere in SQL:

1. l'interrogazione che trova i cognomi di tutti gli utenti che hanno ricevuto esattamente due pacchi da Michele Argento
2. l'interrogazione che trova il pacco più fragile spedito prima di dicembre 2008
3. l'interrogazione che trova tutti i pacchi spediti con ritardo.

Soluzione

1.
```
select cognome, count(*)
from Spedizione, Pacco, Utente u1, Utente u2
where Spedizione.Mittente = u1.IdUtente
and Spedizione.Destinatario = u2.IdUtente
and u1.Nome = 'Michele'
and u1.Cognome = 'Argento'
having count(*) = 2;
```

2.
```
select Pacco.*
from Spedizione S, Pacco P, TipoMerceologico TM
where S.Pacco = Pacco.IdPacco
and TM.IdTipoMerceologico = P.TipoMerceologico
and DataEffettiva < '01-DEC-2008'
and GradoFragilita <= all (
    select GradoFragilita
    from Spedizione S, Pacco P, TipoMerceologico TM
    where S.Pacco = P.IdPacco
    and TM.IdTipoMerceologico = P.TipoMerceologico
    and DataEffettiva < '01-DEC-2008'
    );
```

3.
```
select Pacco.*
from Pacco, Spedizione
where Spedizione.pacco = Pacco.IdPacco and
dataEffettiva > dataStimata.
```