Capitolo 4

Esercizio 4.1

Ordinare i seguenti domini in base al valore massimo rappresentable, supponendo che `integer` abbia una rappresentazione a 32 bit e `smallint` a 16 bit: `numeric(12,4)`, `decimal(10)`, `decimal(9)`, `integer`, `smallint`, `decimal(6,1)`.

Soluzione:

Dominio	Valore Massimo
1. `decimal(10)`	9999999999
2. `integer`	4294967296
3. `decimal(9)`	999999999
4. `numeric(12,4)`	99999999.9999
5. `decimal(6,1)`	99999.9
6. `smallint`	65536

Esercizio 4.2

Definire un attributo che permetta di rappresentare stringhe di lunghezza massima pari a 256 caratteri, su cui non sono ammessi valori nulli e con valore di default “sconosciuto”.

Soluzione:

```
Create domain STRING as character varying (256) default 'sconosciuto'
not null
```

Esercizio 4.3

Dare le definizioni SQL delle tre tabelle

```
FONDISTA(Nome, Nazione, Età)
GAREGGIA(NomeFondista, NomeGara, Piazzamento)
GARA(Nome, Luogo, Nazione, Lunghezza)
```

rappresentando in particolare i vincoli di foreign key della tabella `GAREGGIA`.

Soluzione:

```
Create Table FONDISTA
(
Nome     character(20)      primary key,
Nazione  character(30),
Età      smallint
)

Create table GARA
(
Nome     character(20)     primary key,
Luogo    character(20),
Nazione  character(20),
Lunghezza         integer
)

Create table GAREGGIA
(
NomeFondista      character(20)   references FONDISTA(Nome),
NomeGara          character(20),
Piazzamento       smallint,
primary key (NomeFondista, NomeGara),
foreign key (NomeGara)    references GARA(Nome)
)
```

Esercizio 4.4

Dare le definizioni SQL delle tabelle

```
AUTORE (Nome, Cognome, DataNascita, Nazionalità)
LIBRO (TitoloLibro, NomeAutore, CognomeAutore, Lingua)
```

Per il vincolo foreign key specificare una politica di `cascade` sulla cancellazione e di `set null` sulle modifiche.

Soluzione:

```
Create table AUTORE
(
Nome              character(20),
Cognome           character(20),
DataNascita       date,
Nazionalità       character(20),
primary key(Nome, Cognome)
)

Create table LIBRO
(
TitoloLibro       character(30)    primary key,
NomeAutore        character(20),
CognomeAutore     character(20),
Lingua            character(20),
foreign key (NomeAutore, CognomeAutore)
                  references AUTORE(Nome, Cognome)
                  on delete cascade
                  on update set NULL
)
```

Esercizio 4.5

Dato lo schema dell'esercizio precedente, spiegare cosa può capitare con l'esecuzione dei seguenti comandi di aggiornamento:

```
delete from AUTORE where Cognome = 'Rossi'

update LIBRO set NomeAutore= 'Umberto'
          where CognomeAutore = 'Eco'

insert into AUTORE(Nome,Cognome) values('Antonio','Bianchi')

update AUTORE set Nome = 'Italo' where Cognome = 'Calvino'
```

Soluzione:

1. Il comando cancella dalla tabella AUTORE tutte le tuple con Cognome = 'Rossi'. A causa della politica `cascade` anche le tuple di LIBRO con CognomeAutore = 'Rossi' verranno eliminate.
2. Il comando non è corretto: Nome e Cognome sono attributi della tabella AUTORE e non della tabella LIBRO
3. Il comando aggiunge una nuova tupla alla tabella AUTORE. Non ha alcun effetto sulla tabella LIBRO
4. Le tuple di AUTORE con Cognome = Calvino vengono aggiornate a Nome = Italo. A causa della politica `set null` gli attributi NomeAutore e CognomeAutore delle tuple di Libro con CognomeAutore = Calvino vengono posti a `NULL`.

Esercizio 4.6

Date le definizioni:

```
create domain Dominio as integer default 10
create table Tabella (Attributo Dominio default 5)
```

indicare cosa avviene in seguito ai comandi:

1. `alter table Tabella alter column Attributo drop default`
2. `alter domain Dominio drop default`
3. `drop domain Dominio`

Soluzione:

1. Il comando cancella il valore di default di Attributo. Il valore di default dopo il comando sarà quello impostato in Dominio, ossia 10.
2. Il comando cancella il valore di default di Dominio. Dopo l'operazione il valore di default sarà `NULL`
3. Il comando cancella l'intero dominio Domain. In Tabella il dominio di Attributo diventerà `integer`

Esercizio 4.7 Con riferimento ad una relazione PROFESSORI(CF, Nome, Età, Qualifica), scrivere le interrogazioni SQL che calcolano l'età media dei professori di ciascuna qualifica, nei due casi seguenti:

1. se l'età non è nota si usa per essa il valore nullo
2. se l'età non è nota si usa per essa il valore `0`.

Soluzione

1. Le funzioni aggregative escludono dalla valutazione le ennuple con valori nulli:

```
select Qualifica, avg(Eta) as EtaMedia
from Professori
group by Qualifica
```

2. È necessario escludere esplicitamente dal calcolo della media le ennuple con il valore che denota l'informazione incompleta:

```
select Qualifica, avg(Eta) as EtaMedia
from Professori
where Eta <> 0
group by Qualifica.
```

Esercizio 4.8 Spiegare perché in SQL è previsto (e necessario) un operatore di unione mentre in molte versioni non esistono gli operatori di intersezione e differenza.

Soluzione

In SQL è possibile esprimere l'intersezione attraverso dei join e la differenza attraverso l'uso dell'espressione "NOT IN" mentre non c'è modo di esprimere l'unione.

Esercizio 4.9 Considerare le relazioni IMPIEGATI (Matricola, Nome, Stipendio, Direttore) e DIPARTIMENTI (Codice, Direttore) e le due interrogazioni seguenti, specificare se e in quali casi esse possono produrre risultati diversi:

```
select avg(Stipendio)
from Impiegati
where Direttore in (select Direttore
                    from   Dipartimenti)

select avg(Stipendio)
from Impiegati I, Dipartimenti D
where I.Direttore = D.Direttore .
```

Soluzione
Le due interrogazioni dovrebbero calcolare lo stipendio medio di un direttore di dipartimento.
Nel caso in cui esistono impiegati direttori di più dipartimenti la seconda query produce un risultato scorretto in quanto conta più occorrenze di questi impiegati nel computo della media.
La prima query produce il risultato corretto.

Esercizio 4.10 Si consideri una base di dati sulle relazioni:

- $R_1(\underline{A}, B, C)$
- $R_2(\underline{D}, \underline{E}, F)$.

Facendo riferimento ad una versione dell'SQL che non prevede la differenza (parole chiave `EXCEPT` e `MINUS`) e che permette l'uso dei confronti nella nidificazione solo su singoli attributi (e quindi non ammette condizioni del tipo ... `(A,B) IN SELECT C,D FROM` ...), scrivere interrogazioni in SQL equivalenti alle seguenti espressioni dell'algebra relazionale:

1. $\pi_{BC}(\sigma_{C>10}(R_1))$
2. $\pi_B(R_1 \bowtie_{C=D} \sigma_{F=2}(R_2))$
3. $\pi_{AB}(R_1) - \pi_{AB}(R_1 \bowtie_{C=D} R_2))$.

Soluzione

1. ```
   select distinct B, C
   from R1
   where  C > 10;
   ```

2. ```
   select distinct B
   from R1 join R2
        on (C = D)
   where  F = 2;
   ```

3. ```
   select A, B
   from R1 left join R2
           on (C = D)
   where  C <> D.
   ```
   ```

Esercizio 4.11 Con riferimento alla base di dati nell'esercizio 4.10 scrivere espressioni dell'algebra relazionale equivalenti alle seguenti interrogazioni SQL:

1. ```
   select distinct A, B
   from R1, R2
   where C = D and E > 100;
   ```

2. ```
   select distinct A, B
   from R1 X1
   where not exixts(
       select *
       from R1 Y1, R2
       where Y1.C = D and X1.A = Y1.A and F>10).
   ```

Soluzione

1. $\pi_{AB}(R_1 \rtimes_{C=D} (\sigma_{F>2}(R_2)))$
2. $\pi_{AB}(R_1 - (R_1 \rtimes_{C=D} (\sigma_{F>10}(R_2))))$.

Esercizio 4.12 Con riferimento alla base di dati nell'esercizio 4.10, indicare, per ciascuna delle seguenti interrogazioni, se la parola chiave `DISTINCT` è necessaria:

1. l'interrogazione 1 nell'esercizio 4.11
2. l'interrogazione 2 nell'esercizio 4.11
3. ```
   select distinct A, B
   from R1, R2
   where B = D and C = E
   ```
4. ```
   select distinct B, C
   from R1, R2
   where B = D and C = E.
   ```

Soluzione

1. **SÌ** perché nel join effettuato tra le relazioni R_1 e R_2 sulla condizione C=D potrebbe accadere che una stessa tupla di R_1 si combini con più tuple di R_2 (essendo D solo uno degli attributi della chiave di R_2) generando dei duplicati nel risultato.
2. **NO** perché gli argomenti della select coinvolgono l'attributo A chiave della relazione R_1 e l'operazione di differenza effettuata non produce tuple aggiuntive nel risultato.
3. **NO** perché gli argomenti della select coinvolgono l'attributo A chiave della relazione R_1 e la condizione di join tra le tabelle R_1 e R_2 coinvolge entrambi gli attributi chiave DE della relazione R_2 quindi non potranno essere presenti nel risultato tuple uguali.
4. **SÌ** perché gli argomenti della select non coinvolgono una chiave per la relazione R_1 e sebbene ogni tupla di R_1 si possa combinare con al più una tupla di R_2 a causa delle condizioni di join non è garantito che non ci siano coppie di attributi sempre diversi tra loro su BC.

Esercizio 4.13 Con riferimento ad una base di dati sullo schema R_1(A,B,C), R_2(A,B,C),R_3(C,D,E) considerare l'espressione dell'algebra relazionale $\pi_{AE}((R_1 \cup R_2) \bowtie R_3)$ e scrivere un'espressione SQL ad essa equivalente senza utilizzare il join esplicito (cioè la parola chiave JOIN) né viste.

Soluzione

```
select distinct A, E
from R1, R3
where R1.C = R3.C
union
select A, E
from R2, R3
where R2.C = R3.C.
```

Esercizio 4.14

Dato il seguente schema:

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

scrivere le interrogazioni SQL che permettono di determinare:

1. Le città con un aeroporto di cui non è noto il numero di piste;
2. Le nazioni da cui parte e arriva il volo con codice AZ274;
3. I tipi di aereo usati nei voli che partono da Torino;
4. I tipi di aereo e il corrispondente numero di passeggeri per i tipi di aereo usati nei voli che partono da Torino. Se la descrizione dell'aereo non è disponibile, visualizzare solamente il tipo;
5. Le città da cui partono voli internazionali;
6. Le città da cui partono voli diretti a Bologna, ordinate alfabeticamente;
7. Il numero di voli internazionali che partono il giovedì da Napoli;
8. Il numero di voli internazionali che partono ogni settimana da città italiane (farlo in due modi, facendo comparire o meno nel risultato gli aeroporti senza voli internazionali);
9. Le città francesi da cui partono più di venti voli alla settimana diretti in Italia;
10. Gli aeroporti italiani che hanno solo voli interni. Rappresentare questa interrogazione in quattro modi:
    a) con operatori insiemistici;
    b) con un interrogazione nidificata con l'operatore `not in`;
    c) con un interrogazione nidificata con l'operatore `not exists`;
    d) con l'outer join e l'operatore di conteggio
11. Esprimere l'interrogazione pure in algebra relazionale;
12. Le città che sono servite dall'aereo caratterizzato dal massimo numero di passeggeri;

Soluzione:

1.
```
select Città
from AEROPORTO
where NumPiste is NULL
```

2.
```
select A1.Nazione, A2.Nazione
from AEROPORTO as A1 join VOLO on A1.Città=CittàArr
join AEROPORTO as A2 on CittàPart=A2.Città
where IdVolo= 'AZ274'
```

3.
```
select TipoAereo
from VOLO
where CittàPart='Torino'
```

4.
```
select VOLO.TipoAereo, NumPasseggeri
```

```
        from VOLO left join AEREO
        on  VOLO.TipoAereo=aereo.TipoAereo
        where CittàPart= 'Torino'
```

5.
```
        select CittàPart
        from AEROPORTO as A1 join VOLO on CittàPart=A1.Città
        join AEROPORTO as A2 on CittàArr=A2.Città
        where A1.Nazione <> A2.Nazione
```

6.
```
        select CittàPart
        from VOLO
        where CittàArr= 'Bologna'
        order by CittàPart
```

7.
```
        select count(*)
        from VOLO join AEROPORTO on CittàArr=Città
        where CittàPart = 'Napoli' and Nazione <> 'Italia' and
        GiornoSett= 'Giovedì'
```

8. a.
```
        select count(*), CittàPart
        from AEROPORTO as A1 join VOLO  on A1.Città=CittàPart
        join AEROPORTO as A2 on CittàArr=A2.Città
        where A1.Nazione='Italia' and A2.Nazione <> ' Italia'
        group by CittàPart
```
   b.
```
        select count(CittàArr)
        from AEROPORTO as A1 join VOLO  on A1.Città=CittàPart
        join AEROPORTO as A2 on CittàArr=A2.Città
        where A1.Nazione='Italia' and A2.Nazione <> ' Italia'
        group by CittàPart
```

9.
```
        select CittàPart
        from AEROPORTO as A1 join VOLO on A1.Città=CittàPart
        join AEROPORTO as A2 on CittàArr=A2.Città
        where A1.Nazione='Francia' and A2.Nazione= 'Italia'
        group by CittàPart
        Having count(*) >20
```

10. a.
```
        select CittàPart
        from VOLO join AEROPORTO on CittàPart=Città
        where Nazione = 'Italia'
        except
        select CittàPart
        from AEROPORTO as A1 join VOLO on A1.Città=CittàPart
        join AEROPORTO as A2 on CittàArr=A2.Città
        where (A1.Nazione=' Italia ' and A2.Nazione<>' Italia ' )
```

b.
```
        select CittàPart
        from VOLO join AEROPORTO on CittàPart=Città
        where Nazione= 'Italia' and CittàPart not in
        ( select CittàPart
        from AEROPORTO as A1 join VOLO on
```

```
A1.Città=CittàPart join AEROPORTO as A2 on CittàArr=A2.Città
where  A1.Nazione='Italia' and
A2.Nazione<> 'Italia'  )
```

c.
```
select CittàPart
from VOLO join AEROPORTO as A1 on CittàPart=Città
where Nazione= 'Italia' and
not exists ( select * from VOLO join AEROPORTO as A2
on A2.Città=CittàArr
where A1.Città=CittàPart and
A2.Nazione<>'Italia'  )
```

d.
```
select CittàPart
from AEROPORTO as A1 join VOLO on
A1.Città=CittàPart left join AEROPORTO as A2 on
(CittàArr=A2.Città and A2.Nazione='Italia')
where A1.Nazione='Italia'
group by CittàPart
having count (district A2.Nazione)= 1 )
```

11

$$\Pi_{\text{Cittàpart}}\ \sigma_{\text{Nazione='Italia'}}(\text{AEROPORTO} \rhd\lhd_{\text{Città=CittàPart}} \text{VOLO})$$
$$-$$
$$\Pi_{\text{CittàPart}}\ \sigma_{\text{Nazione='Italia'}}\ (\text{AEROPORTO} \rhd\lhd_{\text{Città=CittàPart}} \text{VOLO} \rhd\lhd_{\text{CittàArr=CittàK}} \rho_{\text{CittàK,NazioneK,nK}\leftarrow\text{Città,Nazione,NumPiste}} (\sigma_{\text{Nazione}\neq\text{'Italia'}}(\text{AEROPORTO})))$$

12
```
select CittàPart
from VOLO join AEREO on VOLO.TipoAereo=AEREO.TipoAereo
where NumPasseggeri=      (select
                          max( NumPasseggeri )from AEREO )
union
select CittàArr
from VOLO join AEREO on VOLO.TipoAereo=AEREO.TipoAereo
where NumPasseggeri=      (select max( NumPasseggeri )
                           from AEREO)
```

Esercizio 4.15

Dato il seguente schema:

```
DISCO(NroSerie, TitoloAlbum,Anno,Prezzo)
CONTIENE(NroSerieDisco,CodiceReg,NroProg)
ESECUZIONE(CodiceReg,TitoloCanz,Anno)
AUTORE(Nome,TitoloCanzone)
CANTANTE(NomeCantante,CodiceReg)
```

formulare le interrogazioni SQL che permettono di determinare:

1. I cantautori (persone che hanno cantato e scritto la stessa canzone) il cui nome inizia per 'D';
2. I titoli dei dischi che contengono canzoni di cui non si conosce l'anno di registrazione;
3. I pezzi del disco con numero di serie 78574, ordinati per numero progressivo, con indicazione degli interpreti per i pezzi che hanno associato un cantante;
4. Gli autori e i cantanti puri, ovvero autori che non hanno mai registrato una canzone e cantanti che non hanno mai scritto una canzone;
5. I cantanti del disco che contiene il maggior numero di canzoni;
6. Gli autori solisti di "collezioni di successi" (dischi in cui tutte le canzoni sono di un solo cantante e in cui almeno tre registrazioni sono di anni precedenti la pubblicazione del disco);
7. I cantanti che non hanno mai registrato una canzone come solisti;
8. I cantanti che non hanno mai inciso un disco in cui comparissero come unici cantanti;
9. I cantanti che hanno sempre registrato canzoni come solisti.

Soluzione:

1.
```
select NomeCantante
from  CANTANTE join ESECUZIONE on
CANTANTE.CodiceReg=ESECUZIONE.CodiceReg
join AUTORE on ESECUZIONE.TitoloCanz=AUTORE.TitoloCanzone
where Nome=NomeCantante and Nome like 'd%'
```

2.
```
select TitoloAlbum
from DISCO join CONTIENE
        on DISCO.NroSerie=CONTIENE.NroSerieDisco
        join ESECUZIONE on
        CONTIENE.CodiceReg=ESECUZIONE.CodiceReg
where ESECUZIONE.anno is NULL
```

3.
```
select NroProg, TitoloCanz, NomeCantante
from (CONTIENE left join CANTANTE on
        CONTIENE.NroSerieDisco=CANTANTE.CodiceReg)
        join ESECUZIONE
        on CONTIENE.codiceReg= ESECUZIONE.CodiceReg
where NroSerieDisco=78574
order by NroProg
```

4.
```
select Nome
from AUTORE
where Nome not in
         ( select NomeCantante
           from CANTANTE )
union
select NomeCantante
from CANTANTE
where NomeCantante not in
      ( select Nome
                    from AUTORE )
```

5.
```
create view DiscoNumerato (NroSerieDisco,NumCanzoni)
as select  NroSerieDisco , count(*)
from CONTIENE
group by NumSerieDisco

select NomeCantante
from CANTANTE join CONTIENE on
                  CANTANTE.CodiceReg=CONTIENE.CodiceReg
                  join DiscoNumerato on

CONTIENE.NroSerieDisco=DiscoNumerato.NroSerieDisco
where NumCanzoni= ( select max (NumCanzoni)
                               from DiscoNumerato)
```

6.
```
select NroSerie
from DISCO
where NroSerie not in
( select NroSerieDisco
  from CONTIENE join CANTANTE as S1 on
  CONTIENE.CodiceReg=S1.CodiceReg
  join CANTANTE as S2 on
  CONTIENE.CodiceReg =S2.CodiceReg
  where   S1.NomeCantante<>S2.NomeCantante )
  and NroSerie in
( select NroSerieDisco
  from (CONTIENE join ESECUZIONE on
  CodiceReg= CodiceReg ) join DISCO on
  DISCO.NroSerie=CONTIENE.NroSerieDisco)
  where ESECUZIONE.Anno<DISCO.Anno
  group by NroSerieDisco
  having count(*) >=3 )
```

7.
```
select distinct NomeCantante
from CANTANTE
where NomeC not in
```

```
        ( select S1.NomeCantante
          from CANTANTE as S1
          where CodiceReg not in
         ( select CodiceReg
           from CANTANTE S2
           where S2.NomeCantante <> S1.NomeCantante ) )
```

8.
```
Create view CantantiUnDisco (NomeCantante) as
select NomeCantante
from CONTIENE join CANTANTE on
         CONTIENE.CodiceReg=CANTANTE.CodiceReg
where NroSerieDisco not in
     (  select NroSerieDisco
        from CONTIENE join CANTANTE as S1 on
                      CONTIENE.CodiceReg=S1.CodiceReg
                       join CANTANTE as S2 on
                      CodiceReg=S2.CodiceReg
        where S1.NomeCantante<>S2.NomeCantante  )

select NomeCantante
from CANTANTE
where NomeCantante not in (select NomeCantante
                           from CantantiUnDisco)
```

9.
```
select NomeCantante
from CANTANTE
where NomeCantante not in
( select S1.NomeCantante
from CANTANTE as S1 join ESECUZIONE on
CodiceReg=S1.CodiceReg join CANTANTE as S2 on
CodiceReg=S2.CodiceReg )
where S1.NomeCantante<> S2.NomeCantante )
```

Esercizio 4.16 Considerare la base di dati relazionale definita per mezzo delle seguenti istruzioni:

```
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)
).
```

Si supponga che vengano registrati anche gli esami non superati, con voti inferiori al 18.
Formulare in SQL:

1. l'interrogazione che trova gli studenti che non hanno superato esami;
2. l'interrogazione che trova gli studenti che hanno riportato in almeno un esame un voto più alto di Archimede Pitagorico;
3. l'interrogazione che trova i nomi degli studenti che hanno superato almeno due esami;
4. l'interrogazione che trova, per ogni studente, il numero di esami superati e la relativa media.

Soluzione

1.
```
select *
from Studenti
where matricola not in (
        select distinct studente
        from Esami);
```

2.
```
select *
from Studenti join Esami
    on matricola=studente
where voto > any (
        select voto
        from Esami join Studenti
                on studente = matricola
                where cognome = 'pitagorico'
                and nome = 'archimede');
```

3.
```
select distinct s.nome, s.cognome
from Studenti s,
     Esami e1 join Esami e2
        on (e1.studente = e2.studente)
where
e1.codiceCorso > e2.codiceCorso and
e1.voto >= 18 and
e2.voto >= 18 and
s.matricola = e1.studente;
```

4.
```
select s.nome, s.cognome, count(*), avg(voto)
from Studenti s, Esami e
where
s.matricola = e.studente
group by s.nome, s.cognome.
```

Esercizio 4.17 Considerare la seguente base di dati relazionale:

- NEGOZI (IDNegozio, Nome, Città)
- PRODOTTI (CodProdotto, NomeProdotto, Marca)
- LISTINO (Negozio, Prodotto, Prezzo) con vincoli di integrità referenziale fra Negozio e la relazione NEGOZI fra Prodotto e la relazione PRODOTTI.

Formulare in SQL:

1. l'interrogazione che fornisce nome e città dei negozi che vendono prodotti della marca XYZ
2. l'interrogazione che trova, per ciascun prodotto, la città in cui viene venduto al prezzo più basso
3. l'interrogazione che trova i prodotti che vengono venduti in una sola città.

Fare riferimento ad una versione dell'SQL che non prevede la differenza (parole chiave EXCEPT e MINUS) e che permette l'uso dei confronti nella nidificazione solo su singoli attributi.
Sono quindi sono ammesse condizioni del tipo

... A IN SELECT C FROM ...

ma non del tipo

... (A,B) IN SELECT C,D FROM ...

Soluzione

1.
```
select distinct Nome, Citta
from Negozi, Prodotti, Listino
where IDNegozio = Negozio
and Prodotto = CodProdotto
and Marca = 'XYZ';
```

2.
```
select distinct P.NomeProdotto, P.Citta
from Negozi, Listino, Prodotti P
where prezzo <= all (
        select Prezzo, NomeProdotto
        from Prodotti P2, Listino
        where P.NomeProdotto = P2.NomeProdotto
        );
```

3.
```
select distinct P1.NomeProdotto
from Prodotti P1
where NomeProdotto not in (
        select P1.NomeProdotto
        from Negozi N1, Negozi N2, Prodotti P2,
        Listino L1, Listino L2
        where N1.IdNegozio = L1.Negozio
        and N2.IdNegozio = L2.Negozio
```

```
and L1.Prodotto = L2.Prodotto
and L1.Prodotto = P2.CodProdotto
and N1.Citta <> N2.Citta).
```

Esercizio 4.18

Dare una sequenza di comandi di aggiornamento che modifichi l'attributo Stipendio della tabella Impiegato, aumentando del 10% gli stipendi sotto i 30 mila € diminuendo del 5% gli stipendi sopra i 30 mila € (gli stipendi di 30.000 € rimangono invariati).

Soluzione:

```
update Impiegato set Stipendio= Stipendio*0.5
where Stipendio < 30000

update Impiegato set  Stipedio= Stipendio*0.95
where Stipendio > 30000

update Impiegato set Stipendio= Stipendio*2.2
where Stipendio < 15000
```

Esercizio 4.19 Considerare la base di dati relazionale con il seguente schema:

- PRODOTTI (Codice, Nome, Categoria)
- VENDITE (CodiceProdotto, Data, Incasso) con vincolo di integrità referenziale fra l'attributo CodiceProdotto e la relazione PRODOTTI

e la sua seguente istanza:

PRODOTTI		
Codice	Nome	Categoria
101	A	Bevanda
102	B	Bevanda
103	C	Pasta
104	D	Biscotti

VENDITE		
CodiceProdotto	Data	Incasso
101	24/11/2008	2.000
101	25/11/2008	1.000
102	23/11/2008	2.500
102	24/11/2008	4.000
103	25/11/2008	1.320

mostrare il risultato delle tre seguenti interrogazioni:

1.
```
select Codice
from Prodotti
where not exists
    (select *
     from Vendite
     where CodiceProd=Codice);
```

2.
```
select Codice
from Prodotti
where not exists
    (select *
     from Vendite
     where Data = 2008-11-24
     and CodiceProd=Codice);
```

3.
```
select Codice
from Prodotti
where not exists
    (select *
     from Vendite
     where Data = 2008-11-24);
```

Soluzione

1.

CODICE
104

2.

CODICE
103
104

3.

CODICE
103

Esercizio 4.20 Con riferimento all'esercizio 4.19 mostrare le istruzioni SQL per creare e popolare l'istanza di basi di dati mostrata.

Soluzione

```
create table Prodotti (
 codice numeric not null primary key,
 nome char(20) not null,
 categoria char(20) not null
);

create table Vendite (
 codiceProd numeric not null
          references Prodotti(codice),
 data date not null,
 incasso numeric not null,
 primary key (codiceProd, data));

delete * from Prodotti;
insert into Prodotti values(101,A,Bevanda);
insert into Prodotti values(102,B,Bevanda);
insert into Prodotti values(103,C,Pasta);
insert into Prodotti values(104,D,Biscotti);

delete * from Vendite;
insert into Vendite values(101,2008-11-24,2000);
insert into Vendite values(101,2008-11-25,1000);
insert into Vendite values(102,2008-11-23,2500);
insert into Vendite values(102,2008-11-24,4000);
insert into Vendite values(103,2008-11-25,1320);
```

Esercizio 4.21 Si consideri una base di dati sulle relazioni:

- R_1 (A,B,C)
- R_2 (D,E, F)

Mostrare unistanza della base di dati che confermi il fatto che le due espressioni seguenti non sono equivalenti:

1. $\pi_{AB}\,(R_1 \bowtie_{B=D} \sigma_{F=2}(R_2))$
2. ```
   select A , B
   from R1, R2
   where B = D
   and C = E
   and F=2.
   ```

Soluzione

R_1		
A	B	C
1	1	1

R_2		
D	E	F
1	2	2

Risultati:

1.

R_1	
A	B
1	1

2.

R_1	
A	B
   ```

Esercizio 4.22 Con riferimento alla base di dati dell'esercizio 4.20, supponendo che le cardinalità delle due relazioni siano rispettivamente N_1 e N_2, indicare le cardinalità (minime e massime) dei risultati delle seguenti interrogazioni:

1.
```
select *
from R1, R2
where C = D
and E > 100;
```

2.
```
select *
from R1 X1
where not exists
    (select *
     from R1 Y1, R2
     where Y1.C = D
     and X1.A = Y1.A
     and F>10);
```

3.
```
select distinct A , B
from R1, R2
where B = D
and C = E.
```

Soluzione

1. Compreso fra 0 e $N_1 \times N_2$
2. Compreso fra 0 e N_1
3. Compreso fra 0 e N_1.

Esercizio 4.23 Considerare la base di dati relazionale definita per mezzo delle seguenti istruzioni:

```
create table Tariffa (
  tipoAuto numeric not null primary key,
  costoAlKm numeric not null
);
create table Automobile (
  targa numeric not null primary key,
  tipologia char(20) not null
        references Tariffa(TipoAuto),
  lunghezza char(20) not null
);
create table Transito (
  codice numeric not null primary key,
  auto numeric not null
        references Automobile(targa),
  orarioIngresso numeric not null,
  orarioUscita numeric not null,
  KmPercorsi numeric not null
).
```

Formulare in SQL:

1. l'interrogazione che restituisce, per ogni transito, i dati del veicolo, del transito e il costo del pedaggio (ottenuto moltiplicando il costo al Km per i Km percorsi);
2. l'interrogazione che restituisce tutti i dati delle automobili che sono transitate più di una volta sull'autostrada;
3. l'interrogazione che restituisce le auto che in ogni transito hanno percorso sempre lo stesso numero di Km;
4. l'interrogazione che restituisce i dati dei transiti (gli stessi richiesti nella prima domanda) per cui la velocità media è superiore ai 140Km/h (si assuma che una differenza tra i due orari produca il risultato espresso in ore).

Soluzione

1.
```
select A.targa, A.tipologia, A.lunghezza,
        T.kmPercorsi * TA.costoAlKm
  from Transito T join Automobile A
       on (T.auto = A.targa)
           join Tariffa TA
               on (TA.tipoAuto = A.tipologia);
```
2.
```
select A.targa, A.tipologia, A.lunghezza
  from Automobile A join Transito T
      on (T.auto = A.targa)
          join transito T2
               on (T1.auto = T2.auto)
  where T1.codice > T2.codice;
```

3.
```
select A.targa
from Automobile A join Transito T
    on (A.targa = T.auto)
        left join Transito T2
            on (T1.auto = T2.auto)
where T1.codice > T2.codice
exept
select A.targa
from Automobile A join Transito T
    on (A.targa = T.auto)
        join Transito T2
            on (T1.auto = T2.auto)
where T1.codice > T2.codice
and (T1.km <> T2.Km);
```
4.
```
select A.*, T.*, T.KmPercorsi * TA.costoKM
from Transito T join Automobile A
    on (T.auto = A.targa)
        join Tariffa TA
            on (TA.tipoAuto = A.Tipologia)
where (T.KmPercorsi / (T.uscita-T.ingresso)) > 140.
```

Esercizio 4.24 Dato uno schema di base di dati relazionale con le seguenti due relazioni:

- R (A,B,C)
- S (C,D,E)

scrivere l'interrogazione in algebra relazionale corrispondente alla seguente interrogazione SQL:

```
select R1.A, R2.C, E
from R R1, R R2, S
where R1.C = R2.A
and S.C = R2.C
and R1.B > 7.
```

Soluzione
$proj_{A,I,E}$ ($\sigma_{B>7}$(R $\bowtie_{C=G}$ (($\rho_{G,H,I \leftarrow A,B,C}$ R) $\bowtie_{I=C}$ S))).

Esercizio 4.25 Si supponga di avere un sistema di basi di dati non in grado di eseguire left join espliciti. Si traduca la query SQL seguente in un suo equivalente supportato da tale dbms.

```
select *
from Animali left join Padroni
     on (padrone = codiceFiscale and etaPadrone>40).
```

Soluzione

```
select *
from Animali, Padroni
where (padrone = codiceFiscale and etaPadrone>40)
or  padrone = null.
```

Esercizio 4.26 Si consideri una base di dati che gestisce dati relativi ai voli in partenza da un dato aeroporto (ad esempio Roma), con le seguenti relazioni:

- AEROPORTI(Codice,Città,Nome);
- AEREI(Codice,Nome,NumeroPosti);
- VOLI(Compagnia,Numero,Destinazione,OraPart,OraArr,Aereo) con vincoli di integrità referenziale fra Destinazione e la relazione AEROPORTI e fra Aereo e la relazione AEREI.

Formulare in SQL:

1. l'interrogazione che trova le città raggiungibili con un volo diretto che utilizzi un aereo con almeno 200 posti.
2. l'interrogazione che trova le città raggiungibili con voli diretti e, per ciascuna, mostra il numero di tali voli.

Soluzione

1.
```
select distinct Citta
from Aeroporti join Voli
     on Aeroporti.Codice = Voli.Destinazione
     join Aerei on Aerei.Codice = Voli.Aereo
where Aerei.NumeroPosti >= 200;
```

2.
```
select Citta, count(*)
from Aereoporti join Voli
on Aeroporti.Codice = Voli.Destinazione
group by Citta.
```