CAPITOLO 4

ESERCIZI SU SQL

(le soluzioni sono riportate da pag. 5 in poi)
(Nota: gli esercizi non sono sempre in ordine di difficoltà)

Esercizio 1

Si prendano tutti gli schemi relazionali considerati negli esercizi sulla progettazione logica pubblicati sulla pagina web del corso. Si diano le definizioni SQL delle relazioni di ciascun schema relazionale, facendo particolare attenzione alle chiavi primarie, ai vincoli intrarelazionali e ai vincoli d'integrità referenziale.

Esercizio 2

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`

Esercizio 3

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.

Esercizio 4

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

1. `delete from AUTORE where Cognome = 'Rossi'`
2. `update LIBRO set Autore= 'Umberto' where Autore = 'Eco'`
3. `insert into AUTORE(Nome,Cognome) values('Antonio','Bianchi')`
4. `update AUTORE set Nome = 'Italo' where Cognome = 'Calvino'`

Esercizio 5

Con riferimento ad una relazione `PROFESSORI(CF, Nome, Eta, 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.

Esercizio 6

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. Le città che sono servite dall'aereo caratterizzato dal massimo numero di passeggeri;

Esercizio 7

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. 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);
6. I cantanti che non hanno mai registrato una canzone come solisti;
7. I cantanti che hanno sempre registrato canzoni come solisti.

Esercizio 8

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.

Esercizio 9

Dare una sequenza di comandi di aggiornamento che modifichi l'attributo Stipendio della tabella Impiegato, aumentando di 200€ gli stipendi sotto i 30000€ e diminuendo del 5% gli stipendi sopra i 30000€ (gli stipendi di 30.000 € rimangono invariati).

Esercizio 10

Considerare la base di dati relazionale con il seguente schema:

- `PRODOTTI (Codice, Nome, Categoria)`
- `VENDITE (CodiceProdotto, Data, Incasso)`
  con vincolo di integrita 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);
   ```

Esercizio 11

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

Esercizio 12

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).

Esercizio 13

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,Citta,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.

SOLUZIONI

Esercizio 2

```
Create Table FONDISTA
(
     Nome  character(20) primary key,
     Nazione  character(30),
     Età integer
)
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) references GARA(Nome),
     Piazzamento integer,
     primary key (NomeFondista, NomeGara)
)
```

Esercizio 3

```
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

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 5

1. Le funzioni aggregative escludono dalla valutazione le ennuple con valori nulli:
```
select Qualifica, avg(Eta) as EtaMedia
from Professori
group by Qualifica
```

2. E' necessario escludere esplicitamente dal calcolo della media le tuple con il valore che denota l'informazione incompleta:
```
select Qualifica, avg(Eta) as EtaMedia
from Professori
where Eta <> 0
group by Qualifica.
```

Esercizio 6

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.
```
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 7

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.
```
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
)
```

6.
```
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 ) )
```

7.
```
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 8

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
and voto > 18
group by s.nome, s.cognome
```

Esercizio 9

1.
```
update Impiegato set Stipendio= Stipendio+200
where Stipendio < 30000
```
2.
```
update Impiegato set Stipedio= Stipendio*0.95
where Stipendio > 30000
```

Esercizio 10

1.

CODICE
104

2.

CODICE
103
104

3.

CODICE
103

Esercizio 11

```
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 12

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
   except
   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 13

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
   ```