Esercizi

Una porzione del database per una compagnia di taxi è caratterizzata dal seguente schema logico relazionale:[1]

SEDI(<u>codice</u>, citta, indirizzo, telefono)
TAXI(<u>targa</u>, sede$_{fk}$, modello, anno_immatricolazione, scadenza_revisione)
AUTISTI(<u>matricola</u>, cognome_nome, telefono, numero_patente, scadenza_patente)
TURNI(<u>taxi</u>$_{fk}$, <u>giorno</u>, <u>orario</u>, autista$_{fk}$)
CORSE(<u>progr</u>, [<u>taxi</u>, <u>giorno</u>, <u>orario</u>]$_{fk(turni)}$, partenza, destinazione, km, importo, contanti$_{bool}$)

Con riferimento allo schema suddetto, esprimere le seguenti interrogazioni in linguaggio SQL (ove non altrimenti specificato).

1. Scrivere l'istruzione DDL per la definizione della relazione CORSE; gli attributi 'km' e 'importo' devono poter memorizzare due cifre decimali; l'attributo 'contanti' è di tipo booleano. Includere, oltre ai vincoli indicati nello schema, il vincolo di ennupla che impone di avere corse con distanza percorsa ed importo strettamente positivi.

2. Aggiungere alla tabella TURNI il vincolo che impedisce ad un autista di effettuare più turni nello stesso giorno.

3. Eliminare dal database tutti i turni che soddisfano le due seguenti condizioni: sono programmati per una data futura; coinvolgono un taxi la cui revisione è scaduta (alla data odierna) oppure coinvolgono un autista la cui patente è scaduta (alla data odierna).

4. Elencare i taxi che non hanno la data di revisione scaduta (rispetto alla data odierna) e, nel corso dell'ultimo mese, hanno effettuato almeno un turno ma non hanno effettuato nessuna corsa di lunghezza superiore a 10 km.

5. Definire la vista relazionale CORSE_AEROPORTO_PER_TURNO, che associa ad ogni turno il numero delle corse effettuate durante quel turno che hanno avuto per destinazione "Aeroporto". Oltre ai dati suddetti, la vista deve includere la matricola e nome dell'autista che ha effettuato il turno, nonché la città della sede del taxi.

6. Elencare le sedi con almeno 10 taxi, ordinandole in base al numero di corse effettuate (a parità di corse, usare l'ordinamento in base al nome della città della sede).

7. Usando l'algebra relazionale, estrarre i dati dei taxi che che hanno effettuato almeno una corsa con partenza o destinazione "Aeroporto" nel corso dell'ultima settimana. Esprimere la stessa interrogazione in SQL.

[1] Un attributo annullabile è indicato con la notazione A*. I vincoli di chiave esterna sono indicati con la notazione A$_{fk}$; la relazione a cui fa riferimento il vincolo è ricavabile facilmente dal nome dell'attributo.

Es: 1)
Scrivere l'istruzione DDL per la definizione della relazione corse; gli attributi 'km' e 'importo' devono poter memorizzare due cifre decimali; l'attributo 'contanti' è di tipo booleano. Includere, oltre ai vincoli indicati nello schema, il vincolo di ennupla che impone di avere corse con distanza percorsa ed importo strettamente positivi.

```
create table corse (
        prog numeric not null,
        taxi varchar(20) not null,
        giorno date not null,
        orario numeric(2) not null,
        partenza varchar(40) not null,
        destinazione varchar(40) not null,
        km numeric(5,2) not null,
        importo numeric(8,2) not null,
        contanti boolean not null,
        foreign key (taxi,giorno,orario) references turni(taxi,giorno,orario),
        primary key (progr, taxi, giorno, orario),
        check( km>0 and importo>0 )
);
```

Es: 2)
Aggiungere alla tabella turni il vincolo che impedisce ad un autista di effettuare più turni nello stesso giorno.

```
alter table turni
add constraint un_turno_al_giorno( giorno, autista );
```

Es: 3)
Eliminare dal database tutti i turni che soddisfano le due seguenti condizioni: sono programmati per una data futura; coinvolgono un taxi la cui revisione è scaduta (alla data odierna) oppure coinvolgono un autista la cui patente è scaduta (alla data odierna).

```
delete from turni(
        where giorno > current_date
-- il 'today' rischia di essere valutato una volta invece current_date viene valutata ad ogni esecuzione
                and (
                        taxi in (  select targa
                                   from taxi
                                   where scadenza revisione < current_date)
                        or
                        autista in ( select matricola
                                     from autisti
                                     where scadenza_patente < current_date)
                )
);
```

Es: 4)
Elencare i taxi che non hanno la data di revisione scaduta (rispetto alla data odierna) e, nel corso dell'ultimo mese, hanno effettuato almeno un turno ma non hanno effettuato nessuna corsa di lunghezza superiore a 10 km.

```
select distinct tx.targa  -- distinct perché ho una join tra tx e tr
from taxi tx, turni tr
where tx.targa = tr.taxi
        and (current_date - tr.giorno) between 1 and 30
        and tx.scadenza_revisione >= current_date
        and not exists(
                select *
                from corse c, turni t2
                where c.taxi = t2.taxi
                        and c.giorno = t2.giorno
                        and c.orario = t2.orario
                        and (current_date - c.giorno) between 0 and 6
                        and t2.taxi = tx.targa
                        and c.km > 10
        );
```

Es: 5)
Definire la vista relazionale corse aeroporto per turno, che associa ad ogni turno il numero delle corse effettuate durante quel turno che hanno avuto per destinazione "Aeroporto". Oltre ai dati suddetti, la vista deve includere la matricola e nome dell'autista che ha effettuato il turno, nonché la città della sede del taxi.

```
create or replace view corse_aeroporto_per_turno
        (taxi, giorno, orario, numero_corse_per_aeroporto, matricola_autista, nome_autista, città_taxi)
as
        select c.taxi, c.giorno, c.orario, count(*), a.matricola, a.cognome_nome, s.citta
        from corse c, turni t, autista a, taxi tx, sedi s
        where c.taxi = t.taxi
                and c.giorno = t.giorno
                and c.orario = t.orario
                and t.autista = a.matricola
                and t.taxi = tx.targa
                and tx.sede = s.codice
                and c.destinazione = 'Aeroporto'
        group by c.taxi, c.giorno, c.orario, a.matricola, a.cognome_nome, s.citta;
```

Es: 6)
Elencare le sedi con almeno 10 taxi, ordinandole in base al numero di corse effettuate (a parità di corse, usare l'ordinamento in base al nome della città della sede).

```
select s.codice, s.città, count(*) as numero_corse,
from taxi tx, sedi s, corse c
where tx.sede = s.codice
        and c.taxi = tx.targa
group by s.codice, s.citta
having count(distinct c.taxi) >= 10
order by numero_corse desc, s.citta
```

Es: 7)
Usando l'algebra relazionale, estrarre i dati dei taxi che che hanno effettuato almeno una corsa con partenza o destinazione "Aeroporto" nel corso dell'ultima settimana. Esprimere la stessa interrogazione in SQL.

```
TX := TAXI
C := SEL_{    (destinazione = 'Aeroporto' OR partenza = 'Aeroporto')
              AND
              0 <= (current_date-giorno) <= 7
          } (CORSE)

PROJ_{TX.*} (
        TX  JOIN_{TX.targa = C.taxi}  C
)
```