Esempio di esercizi per la prima prova in itinere 2012-2013

Questo documento contiene alcuni esercizi di preparazione alla prima prova 2012-2013. Svolgere questi esercizi consente di verificare se si è preparati a sufficienza. Indicativamente, la prova in itinere conterrà esercizi sui seguenti argomenti (la percentuale è indicativa)

- SQL DDL (25%)
- Algebra relazionale e calcolo su tuple (20%)
- SQL DML (55%)

SQL DDL

1. Dato il seguente schema logico e i seguenti vincoli, si scriva il codice SQL che crea le tabelle SQL corrispondenti. Si ricordi di creare le tabelle nell'ordine richiesto dal linguaggio SQL.

   nazioni(id,nome,sigla)
   citta(nome,nazione,numeroAbitanti,mappa,superfice,capitale)
   residenze(citta, nazione,cittadino,inizioResidenza,fineResidenza)
   cittadini(id,nome,cognome,dataNascita)
   distanze(citta1, nazione1,citta2, nazione2,distanza)

   Vincoli integrità referenziale
   citta.nazione -> nazioni.id
   residenze.citta,residenze. nazione-> citta.nome, citta.nazione
   residenze.cittadino -> cittadini.id
   distanze.citta1,distanze.nazione1 -> citta.nome,citta.nazione
   distanze.citta2,distanze.nazione2 -> citta.nome,citta.nazione

   Altri vincoli
   Il sigla è unico per ciascuna nazione ed è lungo esattamente 3 caratteri
   Nessun attributi può essere nullo eccetto: mappa, superfice, fineResidenza, dataNascita.
   Il numero degli abitanti, la superfice e la distanza, se non nulli, devono essere valori positivi.
   La superfice deve essere rappresentata con numeri fino a 99.999,99.
   La distanza deve essere rappresentata con numeri fino a 9.999,99.
   La mappa deve poter contenere un'immagine di dimensione fino a 100M.
   Il campo capitale può assumere i valori 0 o 1.
   Il valore di default di inizioResidenza è la data corrente (è possibile usare la variabile current_date)
   La data di fineResidenza, se non nulla, deve essere successiva a quella di inizio residenza

SQL DML

Dato il seguente schema relazionale

sedi(id,nome,citta)
prodotti(id,nome,sede,prezzo)
difetti(id,prodotto,descrizione,stato)
segnalazioni(difetto,clienteId,clienteNaz,data)
clienti(id,nazione,nome)

con i seguenti vincoli referenziali

prodotti.sede-> sedi.id
difetti.prodotto -> prodotti.id
segnalazioni.difetto ->difetti.id
segnalazioni.clienteId,segnalazioni.clienteNaz -> clienti.id,clientiNaz

Si implementino le seguenti interrogazioni in SQL

1. Trovare tutti i difetti segnalati da “pippo” dopo il “1-1-2000”, restituendo descrizione del difetto, data di segnalazione e nome del prodotto
2. Cercando i difetti di prodotti sviluppati nelle sedi di “Siena”, restituire il nome della sede, il nome del prodotto e la descrizione del difetto. Includere anche il nome delle sedi che non hanno prodotti e il nome dei prodotti che non hanno difetti
3. Ripetere l’interrogazione precedente selezionando solo i difetti nello stato “fixed”. Includere anche il nome delle sedi che non hanno prodotti e il nome dei prodotti che non hanno difetti. (Fare attenzione ai valori nulli nelle colonna stato!)
4. Trovare le coppie di prodotti che hanno lo stesso prezzo, restituendo il nome dei prodotti
5. Trovare il prezzo più alto di un prodotto, restituendo solo il valore di tale massimo
6. Usando l’interrogazione precedente e le sottointerrogazioni, trovare il nome dei prodotti più costosi
7. Definire l’interrogazione 5 come vista e trovare il nome dei prodotti più costosi senza usare sottointerrogazioni
8. Trovare il prezzo medio dei prodotti per sede, restituendo il nome della sede e il prezzo medio
9. Trovare il numero di difetti per prodotto restituendo il nome del prodotto e il numero di difetti. Si considerino solo i difetti diversi da quelli “fixed” e si restituiscano solo i prodotti con più di tre difetti.
10. Si trovi l’id e il nome delle sedi che non sviluppano prodotti che hanno costo maggiore di 100. Implementare questa interrogazione sia con le sottointerrogazioni che con gli operatori insiemistici.
11. Si trovi la media dei difetti per prodotto in ciascuna sede, restituendo il nome della sede e la media

Algebra relazionale e calcolo su tuple
Usando lo schema relazionale usato per le interogazioni in SQL, si scrivano la seguente interrogazioni in algebra relazionale con theta-join e in calcolo su tuple.

1. Trovare tutti i difetti nello stato “fixed”, restituendo, descrizione del difetto, nome del prodotto e nome della sede di produzione del prodotto
2. Trovare il nome delle sedi in cui non si producono prodotti
3. Trovare il nome dei prodotti che hanno prezzo più basso
4. Trovare il nome dei prodotti che hanno almeno 2 difetti
5. Trovare il nome del prodotto che si trovano in seconda posizione quando i prodotti sono ordinati in modo crescente, assumendo che non ci siano prodotti con prezzo uguale (solo calcolo su tuple)

SOLUZIONI

SQL DDL

```
create table nazioni(
        id serial not null primary key,
        nome varchar(20) not null,
        sigla char(3) not null unique
);

create table citta(
        nome varchar(20) not null,
        nazione int not null references nazioni(id),
        numeroAbitanti int  not null check(numeroAbitanti>0),
        mappa blog(100m),
        superfice decimal(7,2) check(superfice is null or superfice>0),
        capitale bit not null,
        primary key(nome,nazione)
);

create table cittadini(
        id serial not null primary key,
        nome varchar(20) not null,
        cognome varchar(20) not null,
        dataNascita date
);

create table residenze(
        citta varchar(20) not null,
        nazione int not null,
        cittadino int not null references cittadini(id),
        inizioResidenza date not null default current_date,
        fineResidenza date check(fineResidenza is null or inizioResidenza<fineResidenza),
        primary key (citta,nazione,cittadino,inizioResidenza),
        foreign key (citta,nazione) references citta(nome,nazione)
);

create table distanze(
        citta1 varchar(20) not null,
        nazione1 int not null,
        citta2 varchar(20) not null,
        nazione2 int not null,
        distanza decimal(6,2) not null check(ditanza>0),
        primary key (citta1,nazione1,citta2,nazione2),
        foreign key (citta1,nazione1) references citta(nome,nazione),
        foreign key (citta2,nazione2) references citta(nome,nazione)
);
```

SQL DML

Per chi volesse provare le interrogazioni, il seguente script permette di creare le tabelle necessarie

```
create table sedi(
    id serial not null primary key,
    nome varchar(10) not null,
    citta varchar(10) );

create table prodotti(
    id serial not null primary key,
    nome varchar(10) not null,
    sede int references sedi(id),
    prezzo int );

create table difetti(
    id serial not null primary key,
    prodotto int references prodotti(id),
    descrizione varchar(20),
    stato varchar(10));

create table clienti(
    id int not null,
    nazione varchar(10) not null,
    nome varchar(20),
    primary key(id,nazione));

create table segnalazioni(
    difetto int not null references difetti(id),
    clienteId int not null,
    clienteNaz varchar(10) not null,
    data date not null,
    foreign key (clienteId, clienteNaz) references clienti(id,nazione),
    primary key(difetto,clienteId,clienteNaz,date))
```

1. Trovare tutti i difetti segnalati da “pippo” dopo il “1-1-2000”, restituendo descrizione del difetto, data di segnalazione e nome del prodotto

```
SELECT   segnalazioni.descrizione, prodotti.nome, segnalazioni.data
FROM   segnalazioni, clienti, difetti, prodotti
WHERE   segnalazioni.clientenaz = clienti.nazione AND
        segnalazioni.clienteid = clienti.id AND
        segnalazioni.difetto = difetti.id AND
        difetti.prodotto = prodotti.id AND
        clienti.nome = 'pippo' AND  data > '2000-1-1'
```

2. Cercando i difetti di prodotti sviluppati nelle sedi di “Siena”, restituire il nome della sede, il nome del prodotto e la descrizione del difetto. Includere anche il nome delle sedi che non hanno prodotti e il nome dei prodotti che non hanno difetti

```
SELECT
  sedi.nome, prodotti.nome,  difetti.descrizione
FROM   sedi LEFT JOIN  prodotti ON sedi.id=prodotti.sede
        LEFT JOIN difetti On prodotti.id = difetti.prodotto
WHERE   sedi.citta = 'siena';
```

3. Ripetere l'interrogazione precedente selezionando solo i difetti nello stato "fixed". Includere anche il nome delle sedi che non hanno prodotti e il nome dei prodotti che non hanno difetti. (Fare attenzione ai valori nulli nelle colonna stato!)

```
SELECT   sedi.nome, prodotti.nome,  difetti.descrizione
FROM
  sedi LEFT JOIN  prodotti ON sedi.id=prodotti.sede
  LEFT JOIN difetti ON prodotti.id = difetti.prodotto
WHERE   sedi.citta = 'siena' AND (difetti.stato IS NULL OR difetti.stato='fixed');
```

4. Trovare le coppie di prodotti che hanno lo stesso prezzo, restituendo il nome dei prodotti

```
SELECT   p1.nome,p2.nome
FROM   prodotti AS p1, prodotti AS p2
WHERE   p1.prezzo = p2.prezzo AND p1.id>p2.id

// la condizione p1.id>p2.id permette di evitare che vengano restituite coppie in cui compare due volte lo
stesso prodotto e, allo stesso, tempo che una coppia sia ripetuta con i prodotti invertiti
```

5. Trovare il prezzo più alto di un prodotto, restituendo solo il valore di tale massimo

```
SELECT max(prezzo)
FROM prodotti
```

6. Usando l'interrogazione precedente e le sottointerrogazioni, trovare il nome dei prodotti più costosi

```
SELECT nome
FROM prodotti
WHERE prezzo = (SELECT max(prezzo) FROM prodotti)
```

7. Definire l'interrogazione 5 come vista e trovare il nome dei prodotti più costosi senza usare sottointerrogazioni

```
CREATE VIEW massimo(prezzo)
AS SELECT max(prezzo) FROM prodotti

SELECT nome
FROM prodotti, massimo
WHERE prodotti.prezzo=massimo.prezzo
```

8. Trovare il prezzo medio dei prodotti per sede, restituendo il nome della sede e il prezzo medio

```
SELECT sedi.nome, AVG(prezzo)
FROM sedi, prodotti
WHERE sedi.id=prodotti.sede
GROUP BY sedi.id, sedi.nome
```

9. Trovare il numero di difetti per prodotto restituendo il nome del prodotto e il numero di difetti. Si considerino solo i difetti diversi da quelli "fixed" e si restituiscano solo i prodotti con più di tre difetti.

```
SELECT prodotti.nome, count(*) as numero
FROM prodotti,difetti
WHERE prodotti.id=difetti.prodotto AND  difetti.stato <> 'fixed'
GROUP BY prodotti.id, prodotti.nome
HAVING count(*)>3
```

10. Si trovi l'id e il nome delle sedi che non sviluppano prodotti che hanno costo maggiore di 100. Implementare questa interrogazione sia con le sottointerrogazioni che con gli operatori insiemistici.

```
SELECT sedi.id, sedi.nome
FROM sedi
```

```
WHERE sedi.id <> all (SELECT prodotti.sede FROM prodotti WHERE prodotti.prezzo>100)
```

oppure

```
SELECT sedi.id, sedi.nome FROM sedi
EXCEPT
SELECT sedi.id, sedi.nome FROM sedi, prodotti
WHERE prodotti.sede=sedi.id AND prodotti.prezzo>100
```

11. Si trovi la media dei difetti per prodotto in ciascuna sede, restituendo il nome della sede e la media

```
CREATE VIEW numeroDifetti(sedeId,sedeNome,numero) AS
    SELECT prodotti.sede, sede.nome,count(*)
    FROM  prodotti,difetti,sede
    WHERE prodotti.id=difetti.prodotto AND sedi.id=prodotti.sede
    GROUP BY prodotti.sede,prodotti.id,prodotti.nome

SELECT sdeNome, AVG(numero) AS media
FROM numeroDifetti
GROUP BY sedeId, sedeNome
```

Algebra relazionale e calcolo su tuple

1. Trovare tutti i difetti nello stato “fixed”, restituendo, descrizione del difetto, nome del prodotto e nome della sede di produzione del prodotto

   - {nomeSede:s.nome, nomeProdotto:p.prodotto, descrizioneDifetto: d.descrizione |
     s(sedi), p(prodotti), d(difetti) |
     s.id=p.sede $\wedge$ p.id=d.prodotto $\wedge$ d.stato=’fixed’ }

   - rSedi=$\rho_{nomeSede,idSede\leftarrow nome,id}$ (sede) //Si rinominano gli attributi che hanno nome uguale per evitare // ambiguità durante i join
     rProdotti=$\rho_{nomeProdotto,idProdotto\leftarrow nome}$ (prodotti)

     $\pi_{nomeSede,nomeProdotto,descrizione}$(rSedi$\rhd\lhd_{idSede=sede}$ rProdotti $\rhd\lhd_{idProdotto=prodotto}$($\sigma_{stato=’fixed’}$ (difetti))

2. Trovare il nome delle sedi in cui non si producono prodotti

   - {s.nome | s(sedi) |
     $\neg\exists$p(prodotto) p.sede=s.id}

     oppure

     {s.nome | s(sedi) |
     $\forall$p(prodotto) p.sede$\neq$s.id }

   - id_sedi_senza_prodotti=$\rho_{sede\leftarrow id}$ (π_{id} (sedi)) - π_{sede} (prodotti))
     π_{nome} (sede $\rhd\lhd_{id=sede}$ id_sedi_senza_prodotti)

3. Trovare il nome dei prodotti che hanno il prezzo più basso

   - {p.nome | p(prodotto) |
     $\neg\exists$p2(prodotto) p.prezzo>p2.prezzo}

     oppure

     {p.nome | p(prodotto) |
     $\forall$p2(prodotto) p.prezzo $\leq$p2.prezzo}

   - prodotti2= $\rho_{id2, nome2,sede2,prezzo2\leftarrow id,nome,sede,prezzo}$(prodotto) // si creano una copia di prodotti
     prodotti_non_minori=$\pi_{id,nome}$(prodotto $\rhd\lhd_{prezzo>prezzo2}$ prodotti2)
     //si confrontano coppie di prodotti per individuare quelli
     //che non possano essere i minori.
     // Poi si proietta sugli attributi che ci interessano
     //(si include id per distinguere prodotti on lo stesso nome

     π_{nome} (($\pi_{id,nome}$ (prodotti) - prodotti_non_minori)
     //La proiezione su prodotti serve a far in modo che la sottrazione operi
     //tabelle con gli stessi attributi

4. Trovare il nome dei prodotti che hanno almeno 2 difetti

- {p.nome | p(prodotti) |
  ∃d(difetti) ∃d2(difetti) d1.id ≠ d2.id ∧d1.prodotto=p.id ∧d2.prodotto=p.id }

- difetti2= $\rho_{id2,prodotto2,descrizione2,stato2 \leftarrow id,prodotto,descrizione,stato}$ (difetti) //si crea una copia di difetti
  id_prodotti_con_due_difetti =π_{id2} (difetti $\triangleright\triangleleft_{id \neq id2 \wedge prodotto=prodotto2}$ (difetti2))
  //Si confrontano coppie di difetti per trovare quelle relative allo stesso prodotto. Si preferisce proiettare su id2 invece di id per evitare ambiguità nel join sottostante

  π_{nome} (prodotti $\triangleright\triangleleft_{id=id2}$ id_prodotti_con_due_difetti) // usando l'id si trova il nome

5. Trovare il nome del prodotto che si trovano in seconda posizione quando i prodotti sono ordinati in modo crescente, assumendo che non ci siano prodotti con prezzo uguale (solo calcolo su tuple)

   - {p.nome | p(prodotti) |
     ∃p2(prodotti) p2.prezzo<p.prezzo ∧ ¬∃p3(prodotti) (p3.id≠p2.id ∧ p3.prezzo<p.prezzo) }