Basi di dati I — Prova di autovalutazione 4 novembre 2013
Soluzioni

Domanda 1 (25%) Si consideri una base di dati sulle relazioni

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

Scrivere interrogazioni in SQL equivalenti alle seguenti espressioni dell'algebra relazionale:

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

Possibile soluzione

1. ```
   SELECT DISTINCT B , C
   FROM R1
   WHERE C > 10
   ```
2. ```
   SELECT DISTINCT B
   FROM R1, R2
   WHERE C = D AND F = 2
   ```

Domanda 2 (25%) Con riferimento alla base di dati nella domanda 1 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 EXISTS
   (SELECT *
   FROM R1 Y1, R2
   WHERE Y1.C = D AND X1.A = Y1.A AND F>10)
   ```

Possibile soluzione

1. $\pi_{AB}(R_1 \bowtie_{C=D} \sigma_{E>100}(R_2))$
2. $\pi_{AB}(R_1) - \pi_{AB}(R_1 \bowtie_{C=D} \sigma_{F>10}(R_2))$
   Nota: le proiezioni si possono fare direttamente su AB perché A è chiave, e quindi ad ogni valore di A è associato un solo valore di B; se così non fosse stato, le due proiezioni avrebbero dovuto essere su A e poi sul risultato della differenza si sarebbe dovuto eseguire un join con R_1 (in particolare, un join naturale oppure un equi-join preceduto da una ridenominazione): $\pi_{AB}(R_1 \bowtie (\pi_A(R_1) - \pi_A(R_1 \bowtie_{C=D} \sigma_{F>10}(R_2))))$
   Per vedere un esempio che illustri la situazione, considerare la base di dati seguente (che fa riferimento ad uno schema su cui A non è chiave):

A	B	C
1	2	2
1	1	1

D	E	F
2	3	12
1	5	9

   Il risultato dell'espressione $\pi_{AB}(R_1) - \pi_{AB}(R_1 \bowtie_{C=D} \sigma_{F>10}(R_2))$ è

A	B
1	1

   mentre il risultato dell'interrogazione SQL proposta è la relazione vuota su AB. Il risultato dell'espressione $\pi_{AB}(R_1 \bowtie (\pi_A(R_1) - \pi_A(R_1 \bowtie_{C=D} \sigma_{F>10}(R_2))))$ è effettivamente la relazione vuota

Domanda 3 (15%) Ancora con riferimento alla base di dati nella domanda 1, indicare, per ciascuna delle seguenti interrogazioni, se la parola chiave `DISTINCT` è necessaria

1. l'interrogazione 1 nella domanda 2
2. l'interrogazione 2 nella domanda 2
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
```

Possibile soluzione

1. **SÌ**
2. **NO**
3. **NO**
4. **SÌ**

Domanda 4 (35%) 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,
   età 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 hanno riportato in almeno un esame un voto più alto di quello ottenuto, in tale esame, da uno studente di cognome Rossini
2. l'interrogazione che trova, per ciascuno studente, il numero di esami superati e la relativa media
3. l'interrogazione che trova, per ogni corso, lo studente che ha preso il voto più alto
4. l'interrogazione che trova gli studenti che non hanno superato esami (proporre due soluzioni, una che utilizza l'operatore `EXCEPT` e l'altra che non lo utilizza)
5. l'interrogazione che trova, per ogni corso, il numero di studenti che hanno superato l'esame e il numero di quelli che hanno ottenuto, in tale corso, un voto più basso della propria media (in modo informale ma chiaro: si vogliono trovare i corsi che "abbassano la media")

Possibile soluzione

Può essere utile la vista:

```
create view esamiStudOK
as select *
   from studenti join esami on matricola = studente
   where voto >= 18
```

1. Supponendo si tratti di esami superati:
```
select s.matricola, s.cognome, s.codiceCorso, s.voto, r.voto as votoRossini
from esamiStudOK s, esamiStudOK r
where r.cognome='Rossini'
  and s.codiceCorso = r.codiceCorso and s.voto > r.voto
```
2. Usuale interrogazione con aggregazione
```
SELECT matricola, cognome, nome, count(*) as numEsami, avg(voto) as media
from esamiStudOK
```

per eliminare i decimali (in alcuni sistemi)

```
select matricola, cognome, nome, count(*) as numEsami,
      cast(avg(voto) as numeric(4,1)) as media
from esamiStudOK
group by matricola, cognome, nome
```

3. secondo lo standard

```
select matricola, cognome, nome, codiceCorso, voto
from esamiStudOK e
where voto >= all (select voto
                   from esami
                   where codiceCorso = e.codiceCorso)
```

oppure

```
select matricola, cognome, nome, codiceCorso, voto
from esamiStudOK e
where voto = (select max(voto)
              from esami
              where codiceCorso = e.codiceCorso)
```

su alcuni sistemi (ad esempio MySQL) si può scrivere anche nel modo seguente (scosigliato, appunto perché non standard)

```
select matricola, cognome, nome, codiceCorso, max(voto)
from esamiStudOK e
group by codiceCorso
```

4. nei sistemi che prevedono la differenza

```
select matricola , cognome, nome
from studenti
except
select matricola , cognome, nome
from esamiStudOK
```

altrimenti

```
select matricola , cognome, nome
from studenti s
where not exists (
 select *
 from esamistudOK e
 where s.matricola = e.matricola)
```

5. utilizziamo diverse viste

```
create view studMedia (studente, media)
as select studente , avg(voto)
   from esami
   where voto >= 18
   group by studente

create view votiBassiPerCorso as
 select codiceCorso, count(*) as votiBassi
 from esamiStudOK e , studMedia s
 where e.studente = s.studente
   and voto < media
 group by codiceCorso

create view esamiPerCorso as
 select codiceCorso, count(*) as numeroEsami
 from esamiStudOK
 group by codiceCorso

select e.codiceCorso, numeroEsami, votiBassi
from esamiPerCorso e , votiBassiPerCorso v
where e.codiceCorso = v.codiceCorso
```

La soluzione mostrata non considera gli esami in cui nessuno studente ha preso un voto più basso della propria media; per includerli si può usare un'altra vista ancora:

```
create view votiBassiPerCorsoConZero as
 select codiceCorso, count(*) as votiBassi
```

```
 from esamiStudOK e , studMedia s
 where e.studente = s.studente
   and voto < media
 group by codiceCorso
 union
 select codiceCorso, 0 as votiBassi
  from esamiStudOK
  where codiceCorso not in (
    select codiceCorso
  from votiBassiPerCorso) ;

select e.codiceCorso, numeroEsami, votiBassi
from esamiPerCorso e , votiBassiPerCorsoConZero v
where e.codiceCorso = v.codiceCorso ;
```