Interrogazioni in SQL

Corso di Basi di Dati

4 dicembre 2013

1 Interrogazioni sullo schema aziendale

1. Ottenere i nomi dei dipartimenti dislocati in piú sedi.
2. Ottenere il numero di supervisori.
3. Produrre in output l'elenco dei nomi di tutti dipartimenti con il corrispondente numero di sedi, ordinato in modo crescente rispetto al numero di sedi.
4. Ottenere i dati degli impiegati che non sono supervisori.
5. Ottenere i nomi degli impiegati che lavorano almeno 20 ore in ciascuno dei progetti in cui sono impiegati.
6. Ottenere i nomi dei manager che guadagnano piú di ogni impiegato che lavora al progetto 7.

2 Interrogazioni sullo schema scacchistico

Si consideri il seguente schema di base di dati:

GIOCATORE(nome: stringa, nazionalità: stringa)

APERTURA(eco: dom_eco, nome: stringa, variante: stringa)
VNN: {nome}
UNI: {variante}

PARTITA(bianco: stringa, nero: stringa, luogo: stringa, anno: $\mathbb{N}$, round: $\mathbb{N}$, eco: dom_eco, risultato: dom_risultato, num_mosse: $\mathbb{N}$)
CE: bianco → GIOCATORE
CE: nero → GIOCATORE
CE: eco → APERTURA

Si risolvano le seguenti interrogazioni in SQL:

7. Ottenere i nomi di tutti i giocatori che hanno giocato qualche partita.
8. Ottenere i nomi dei giocatori russi che hanno giocato con il Nero contro Kasparov.
9. Ottenere i codici delle aperture che non sono state mai giocate.
10. Determinare le nazionalità dei giocatori che hanno giocato con il Bianco.
11. Determinare le possibili coppie di giocatori di nazionalità diverse.
12. Ottenere le varianti di aperture giocate a Filadelfia.
13. Ottenere i nomi di aperture giocate da Kasparov.
14. Ottenere il numero di mosse di partite aperte con la Difesa Siciliana.

15. Ottenere le coppie di città tali che esiste una partita giocata nella seconda città almeno dieci anni dopo una partita giocata nella prima città.

16. Ottenere i nomi dei giocatori che (col Bianco) giocano partite di Donna, cioè aprono con '1 d4' (suggerimento: usare l'operatore **like** in una condizione del tipo

    ... **where** variante **like** '1 d4%' ...;

    il carattere % corrisponde a qualunque stringa di zero o piú caratteri).

17. Ottenere i nomi delle aperture non giocate da Karpov col Bianco.

18. Ottenere i nomi e le varianti delle aperture non giocate da giocatori russi.

19. Ottenere i risultati di tutte le partite il cui risultato è noto e il nome della corrispondente apertura.

Possibili soluzioni

Schema aziendale

1
```
-- Ottenere i nomi dei dipartimenti dislocati in piú sedi.
select distinct D.dnome from dipartimento D, sedi_dipartimento SD1, sedi_dipartimento SD2
where D.dnumero = SD1.dnumero and D.dnumero = SD2.dnumero
and SD1.dsede <> SD2.dsede;
```

2 La soluzione corretta è:

```
select count(distinct supervisore) from impiegato;
```

La parola chiave **distinct** è necessaria per non contare piú volte lo stesso supervisore.

3
```
select D.dnome, count(dsede) as num_sedi
from sedi_dipartimento SD right outer join dipartimento D
on SD.dnumero = D.dnumero
group by D.dnumero
order by num_sedi asc;
```

La join esterna consente di tener conto dei dipartimenti che non hanno una sede associata. Per tali dipartimenti, la funzione **count**(dsede) produce correttamente il valore 0.

4 In prima battuta si potrebbe pensare a questa soluzione:

```
select * from impiegato
  where cf not in (
    select supervisore from impiegato
);
```

Tuttavia, poiché il campo supervisore ammette valori nulli, l'interrogazione non restituisce alcun record (perché?). Una soluzione corretta è la seguente:

```
select * from impiegato
  where cf not in (
    select supervisore from impiegato
      where supervisore is not null
);
```

In alternativa:

```
select * from impiegato I
  where not exists (
    select * from impiegato S
      where S.supervisore = I.cf
);
```

o anche

```
select * from impiegato
where cf in (
  select cf from impiegato
    except
  select supervisore from impiegato
);
```

5 L'interrogazione può essere riformulata in modo (quasi) equivalente come segue: "ottenere i nomi degli impiegati tali che non esiste alcun progetto in cui lavorano meno di 20 ore". Una prima implementazione (non corretta) è dunque la seguente:

```
select I.nome, I.cognome from Impiegato I -- nome di ciascun impiegato...
where not exists ( -- ...tale che non esiste...
  select * from Lavora_a L -- ...alcun progetto...
    where L.imp = I.cf -- ...in cui tale impiegato è occupato...
      and L.ore_settimana < 20 -- ...e in cui lavora meno di 20 ore.
);
```

Putroppo gli impiegati non assegnati ad alcun progetto sono inclusi nel risultato dell'interrogazione. L'interrogazione può essere corretta come segue:

```
select I.nome, I.cognome from Impiegato I -- Come sopra...
where not exists (
  select * from Lavora_a L
    where L.imp = I.cf
      and L.ore_settimana < 20
)
and exists ( -- ...ed esiste...
  select * from Lavora_a L -- ...almeno un progetto...
    where L.imp = I.cf -- ...in cui l'impiegato è occupato.
);
```

Quest'ultima interrogazione non è tuttavia ancora corretta se la colonna ore_settimana può assumere valori nulli. Se infatti modifichiamo la base di dati come segue:

```
alter table lavora_a alter column ore_settimana drop not null;
insert into lavora_a(imp, progetto, ore_settimana)
  values ('MRRPML65G07R697A', 1, null);
```

l'impiegato MRRPML65G07R697A sarà presente nel risultato della precedente interrogazione, perché la condizione L.ore_settimana < 20 per tale impiegato ha valore di verità indefinito e quindi il record corrispondente non è incluso nel risultato dell'interrogazione annidata. In tal caso, l'interrogazione va riscritta come segue:

```
select I.nome, I.cognome from Impiegato I -- Come sopra...
where not exists (
  select * from Lavora_a L
    where L.imp = I.cf
      and (L.ore_settimana < 20 or L.ore_settimana is null)
)
and exists (
  select * from Lavora_a L
    where L.imp = I.cf
);
```

Un'altra possibile formulazione dell'interrogazione, apparentemente corretta, è la seguente:

```
select cf, nome, cognome from Impiegato -- nome di ciascun impiegato...
where cf not in ( -- ...che non appartiene all'insieme...
  select imp from Lavora_a -- ...degli impiegati che lavorano a un progetto...
   where ore_settimana < 20 -- ...per meno di 20 ore.
);
```

Quest'ultima interrogazione ha lo stesso problema della prima (include anche gli impiegati che non sono assegnati ad alcun progetto), perciò si può pensare di modificarla come segue:

```
select cf, nome, cognome from Impiegato -- Come sopra
where cf not in (
  select imp from Lavora_a
   where ore_settimana < 20
)
and cf in (select imp from Lavora_a); -- cf è assegnato a un progetto.
```

Questa versione è corretta se ore_settimana non ammette valori nulli. In caso contrario, bisogna modificare la clausola **where**:

```
select cf, nome, cognome from Impiegato
where cf not in (
  select imp from Lavora_a
   where ore_settimana < 20 or ore_settimana is null
)
and cf in (select imp from Lavora_a);
```

Infine, vale la pena sottolineare che, nel caso di un impiegato di cui non è noto il monte ore, la risposta corretta alla domanda "lavora al progetto almeno n ore?" è "può darsi". Le soluzioni proposte tuttavia riportano nel risultato soltanto i record per cui le condizioni imposte sono soddisfatte *con certezza*.

6 Questa sembra una soluzione corretta:

```
select cf, nome, cognome, stipendio from impiegato
  where cf in (select manager from dipartimento)
    and stipendio > all (
      select stipendio
        from impiegato join lavora_a on cf = imp
        where progetto = 7
);
```

e infatti restituisce il risultato corretto nell'istanza della base di dati aziendale fornita agli studenti. Se si inseriscono tuttavia i seguenti dati:

```
insert into impiegato(cf, nome, cognome)
  values ('AAAAAAAAAAAAAAAA', 'Ugo', 'Fantozzi');
insert into lavora_a(imp, progetto, ore_settimana)
  values ('AAAAAAAAAAAAAAAA', 7, 100);
```

l'interrogazione precedente produce la tabella vuota (perché?). Una soluzione corretta è la seguente:

```
select cf, nome, cognome, stipendio from impiegato M
  where M.cf in (select manager from dipartimento)
    and not exists (
      select * from impiegato join lavora_a on cf = imp
        where progetto = 7
          and stipendio >= M.stipendio
);
```

Schema scacchistico

Nel seguito si assume che lo schema sia stato creato come segue:

```
create schema Scacchi;

create domain Scacchi.dom_eco as char(3);

create domain Scacchi.dom_risultato as char(3)
  check (value = '1−0' or value = '0−1' or value = '1/2');

create domain Scacchi.dom_nome_giocatore as varchar(20);

create table Scacchi.Giocatore (
  nome Scacchi.dom_nome_giocatore primary key,
  nazionalita varchar(20)
);

create table Scacchi.Apertura (
  eco Scacchi.dom_eco primary key,
  nome varchar(60) not null,
  variante varchar(256)
);

create table Scacchi.Partita (
  bianco Scacchi.dom_nome_giocatore references Scacchi.Giocatore
            on update cascade on delete restrict,
  nero Scacchi.dom_nome_giocatore references Scacchi.Giocatore
            on update cascade on delete restrict,
  luogo varchar(30),
  anno integer check (anno > 1600), -- Solo partite 'moderne'
  round integer check (round > 0),
  eco Scacchi.dom_eco references Scacchi.Apertura
            on update cascade on delete restrict,
  risultato Scacchi.dom_risultato,
  num_mosse integer check (num_mosse >= 0),
  primary key(bianco, nero, luogo, anno, round)
);
```

7

```
-- Nomi di tutti i giocatori che hanno giocato qualche partita
select bianco from Scacchi.Partita
union
select nero from Scacchi.Partita;
```

8

```
-- Nomi di giocatori russi che hanno giocato con il Nero contro Kasparov
select nome from Scacchi.Giocatore where nazionalita = 'Russia'
intersect
-- Avversari che hanno giocato con il Nero contro Kasparov
select nero from Scacchi.Partita where bianco = 'Kasparov';

-- Soluzione alternativa:
select distinct G.nome from Scacchi.Giocatore G, Scacchi.Partita P
where G.nazionalita = 'Russia' -- il giocatore è russo...
and G.nome = P.nero -- ...ha giocato col nero...
and P.bianco = 'Kasparov' -- ...e il suo avversario era Kasparov
```

9

```
-- Codici delle aperture che non sono state mai giocate
select eco from Scacchi.Apertura -- codici di tutte le aperture
except
select eco from Scacchi.Partita; -- codici delle aperture giocate

-- Soluzione alternativa
select eco from Scacchi.Apertura natural left join Scacchi.Partita
where luogo is null;
```

10

```
-- Nazionalità dei giocatori che hanno giocato con il Bianco
select distinct G.nazionalita from
  Scacchi.Giocatore G join Scacchi.Partita P on G.nome = P.bianco;

-- Soluzione alternativa:
select distinct G.nazionalita from Scacchi.Giocatore G, Scacchi.Partita P
  where G.nome = P.bianco;
```

11

```
-- Coppie di giocatori di nazionalità diverse
select G1.nome, G2.nome from Scacchi.Giocatore G1, Scacchi.Giocatore G2
where G1.nazionalita < G2.nazionalita -- '<', invece di '<>', elimina la ridon-
danza
order by G1.nome, G2.nome; -- Ordina il risultato

-- Soluzione alternativa:
select G1.nome, G2.nome
  from Scacchi.Giocatore G1 join Scacchi.Giocatore G2
    on G1.nazionalita < G2.nazionalita;

-- Se si vuole tener conto di Deep Blue (che non ha una nazionalita associata):
select G1.nome, G2.nome from Scacchi.Giocatore G1, Scacchi.Giocatore G2
where G1.nazionalita < G2.nazionalita
or (G1.nazionalita is not null and G2.nazionalita is null)
order by G1.nome, G2.nome;
```

12

```
-- Varianti di aperture giocate a Filadelfia
select variante from Scacchi.Apertura natural join Scacchi.Partita
  where luogo = 'Filadelfia';

-- Soluzione alternativa:
select A.variante from Scacchi.Apertura A, Scacchi.Partita P
  where P.luogo = 'Filadelfia' and A.eco = P.eco;
```

13

```
-- Nomi di aperture giocate da Kasparov
select nome from Scacchi.Apertura natural join Scacchi.Partita
  where bianco = 'Kasparov';

-- Soluzione alternativa:
select A.nome from Scacchi.Apertura A, Scacchi.Partita P
 where A.eco = P.eco and P.bianco = 'Kasparov';
```

14

```
-- Numero di mosse di partite aperte con la Difesa Siciliana
select num_mosse from Scacchi.Partita natural join Scacchi.Apertura
  where nome = 'Difesa Siciliana';
```

```
-- Soluzione alternativa:
select P.num_mosse from Scacchi.Partita P, Scacchi.Apertura A
  where P.eco = A.eco and A.nome = 'Difesa Siciliana';
```

15

```
-- Coppie di città tali che esiste una partita giocata nella seconda città
-- almeno dieci anni dopo una partita giocata nella prima città
select distinct P1.luogo, P2.luogo from Scacchi.Partita P1 join Scacchi.Partita P2
  on P2.anno - P1.anno >= 10;

-- Se si vogliono escludere dal risultato le coppie di città uguali
-- (città in cui si sono giocate partite a distanza di almeno dieci anni):
select distinct P1.luogo, P2.luogo from Scacchi.Partita P1 join Scacchi.Partita P2
  on P2.anno - P1.anno >= 10 and P1.luogo <> P2.luogo; -- qui non si deve
usare <
```

16

```
-- Giocatori che giocano partite di Donna
select P.bianco from Scacchi.Partita P natural join Scacchi.Apertura A
  where A.variante like '1 d4%';

-- Soluzione alternativa:
select P.bianco from Scacchi.Partita P, Scacchi.Apertura A
  where A.variante like '1 d4%' and P.eco = A.eco;
```

17

```
-- Nomi di aperture non giocate da Karpov
select nome from Scacchi.Apertura -- nomi di tutte le aperture
except
select nome from Scacchi.Apertura natural join Scacchi.Partita
  where bianco = 'Karpov'; -- nomi di aperture giocate da Karpov col Bianco

-- Soluzione alternativa
select distinct A.nome
from Scacchi.Apertura A natural left join Scacchi.Partita P
where bianco <> 'Karpov'
or bianco is null; -- aperture mai giocate
```

18

```
-- Nomi e varianti di aperture non giocate da giocatori russi
select nome, variante from Scacchi.Apertura -- Nomi e varianti di tutte le aper-
ture
except
-- Nomi e varianti di aperture giocate da giocatori russi
select A.nome, A.variante
from (Scacchi.Apertura A natural join Scacchi.Partita P)
      join Scacchi.Giocatore G on (G.nome = P.bianco or G.nome = P.nero)
where G.nazionalita = 'Russia';

-- Soluzione alternativa
select A.nome, A.variante
from (Scacchi.Partita P join Scacchi.Giocatore G
  on (G.nome = P.bianco or G.nome = P.nero) and G.nazionalita = 'Russia')
  right join Scacchi.Apertura A on A.eco = P.eco
where G.nome is null;
```

```
19  -- Risultati di tutte le partite il cui risultato è noto e nomi delle corrispondenti
    aperture
    select P.risultato, A.nome
    from Scacchi.Apertura A natural right join Scacchi.Partita P
    where P.risultato is not null;
```