CSE 132A

Solutions to practice problems on tuple calculus and SQL

Problem 1

(a) List the bars that serve a beer that Joe likes.

(i) tuple calculus:

$$\{b : bar \mid \exists s \in serves \exists l \in likes$$
$$(s(bar) = b(bar) \wedge s(beer) = l(beer) \wedge l(drinker) = Joe)\}.$$

(ii) SQL:

select s.bar
from serves s, likes l
where s.beer = l.beer
AND l.drinker = “Joe”

(b) List the drinkers that frequent at least one bar that serves a beer they like.

(i) tuple calculus:

$$\begin{aligned}&\{d : drinker \mid \exists f \in frequents \exists s \in serves \exists l \in likes\\&(d(drinker) = f(drinker) \wedge f(bar) = s(bar)\\&\wedge s(beer) = l(beer) \wedge l(drinker) = f(drinker))\}\end{aligned}$$

(ii) SQL:

select f.drinker
from frequents f, serves s, likes l
where f.bar = s.bar
and s.beer = l.beer
and l.drinker = f.drinker

(c) List the drinkers that frequent only bars that serve some beer that they like.
(Assume each drinker likes at least one beer and frequents at least one bar.)

(i) tuple calculus:

$$\{d : drinker \mid \exists f \in frequents\ (f(drinker) = d(drinker) \land$$
$$\forall y \in frequents[y(drinker) = f(drinker) \rightarrow$$
$$\exists s \in serves \exists l \in likes(s(bar) = y(bar) \land$$
$$s(beer) = l(beer) \land l(drinker) = y(drinker))]\}$$

Existential form:

$$\{d : drinker \mid \exists f \in frequents\ (f(drinker) = d(drinker) \land$$
$$\neg\exists y \in frequents[y(drinker) = f(drinker) \land$$
$$\neg\exists s \in serves \exists l \in likes(s(bar) = y(bar) \land$$
$$s(beer) = l(beer) \land l(drinker) = y(drinker))]\}$$

(ii) SQL:

Using NOT EXISTS (see tuple calculus query above):

```
select f.drinker
from frequents f
where not exists
        (select *
        from frequents y
        where y.drinker = f.drinker and not exists
                (select *
                from serves s, likes l
                where s.bar= y.bar
                and s.beer= l.beer
                and l.drinker = y.drinker))
```

Another version using NOT IN:

```
select drinker
from frequents  where drinker not in
        (select f.drinker
        from frequents f
        where f.bar not in
                (select bar
                from serves, likes
                where serves.beer = likes.beer
                and likes.drinker = f.drinker))
```

(d) List the drinkers who frequent no bar that serves a beer that they like.

This is just the complement of (b).

(e) List the drinkers such that every bar they frequent serves every beer they like.

(i) relational calculus:

$$\{d : drinker \mid \exists f \in frequents(f(drinker) = d(drinker) \wedge$$
$$\forall x \in frequents\, \forall l \in likes\, ((x(drinker) = f(drinker) \wedge l(drinker) = f(drinker)) \rightarrow$$
$$\exists s \in serves\ (s(bar) = x(bar)\ \wedge\ s(beer) = l(beer))))\}$$

Existential form:

$$\{d : drinker \mid \exists f \in frequents(f(drinker) = d(drinker) \wedge$$
$$\neg\exists x \in frequents\, \exists l \in likes\, (x(drinker) = f(drinker) \wedge l(drinker) = f(drinker) \wedge$$
$$\neg\exists s \in serves\ (s(bar) = x(bar)\ \wedge\ s(beer) = l(beer))))\}$$

SQL:

```
select f.drinker from frequents f
where not exists
(select * from frequents x, likes l
where x.drinker = f.drinker and l.drinker = f.drinker and not exists
(select * from serves s
where s.bar = x.bar and s.beer = l.beer))
```

Problem 2

(c) List the actors cast only in movies by Berto.

(i) tuple calculus:

$$\{a : actor \mid \exists m \in movie[a(actor) = m(actor) \wedge$$
$$\forall t \in movie\ (t(actor) = m(actor) \rightarrow\ \exists s \in movie\ (s(title) = t(title)$$
$$\wedge\ s(director) = Berto))]\}$$

EXISTENTIAL form:

$$\{a : actor \mid \exists m \in movie[a(actor) = m(actor) \wedge$$
$$\neg\exists t \in movie\ (t(actor) = m(actor)\ \wedge\ \neg\exists s \in movie\ (s(title) = t(title)$$
$$\wedge\ s(director) = Berto))]\}$$

(ii) SQL (direct translation of the above calculus query, using NOT EXISTS):

```
select m.actor
from movie m
where not exists
        (select * from movie t
        where t.actor = m.actor and not exists
                (select * from movie s
                where s.title = t.title and s.director = 'Berto'))
```

Another possibility, making the unique director assumption:

```
select actor
from movie
where actor not in
        (select actor
        from movie
        where director ≠ Berto )
```

(b) List all pairs of distinct actors who act together in at least one movie (avoid listing both (a, b) and (b, a)).

(i) tuple calculus:

$$\{a : actor1, actor2 \mid \exists m1 \in movie\ \exists m2 \in movie(a(actor1) = m1(actor)$$

$$\wedge\ a(actor2) = m2(actor) \wedge m1(title) = m2(title) \wedge m1(actor) < m2(actor))\}$$

(ii) SQL:

select m1.actor **as** actor1, m2.actor **as** actor2
from movie m1, movie m2
where m1.title = m2.title **and** m1.actor < m2.actor

(c) List the directors such that every actor is cast in one of his/her movies.

(i) tuple calculus (no unique director assumption):

$$\{d : director \mid\ \exists m \in movie\ [d(director) = m(director)\ \wedge$$

$$\forall t \in movie\ \exists z \in movie(z(actor) = t(actor) \wedge z(director) = m(director))]\}$$

EXISTENTIAL form:

$$\{d : director \mid\ \exists m \in movie\ [d(director) = m(director)\ \wedge$$

$$\neg\exists t \in movie\ \neg\exists z \in movie(z(actor) = t(actor) \wedge z(director) = m(director))]\}$$

(ii) SQL (direct translation of the above calculus query):

```
select m.director from movie m
where not exists
        (select * from movie t
        where not exists
               (select * from movie z
               where z.actor = t.actor and z.director = m.director
```

Another possibility:

```
select director
from movie
where director not in
        (select f.director
        from movie f, movie g
        where f.director not in
                (select director
                from movie
                where actor = g.actor ))
```