**CSE 232** **Fall, 2000**

# Solutions to practice problems on queries

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) domain calculus:

$$\{b : bar \mid \exists c : beer(serves(b, c) \wedge likes(c, Joe))\}.$$

   (iii) relational algebra:

$$\pi_{bar}(serves \bowtie \sigma_{drinker=Joe}(likes)).$$

   (iv) SQL:

**select** *serves.bar*
**from** *serves*, *likes*
**where** *serves.beer* = *likes.beer*
AND *likes.drinker* = "Joe"

   (v) QBE:

| *serves* | *bar* | *beer* |
|---|---|---|
| | _b | _c |

| *likes* | *drinker* | *beer* |
|---|---|---|
| | *Joe* | _c |

| *answer* | *bar* |
|---|---|
| **I.** | _b |

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

   (i) tuple calculus:

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

(ii) domain calculus:

$$\{d : drinker \mid \exists b \exists c[frequents(d, b) \land serves(b, c) \land likes(d, c)]\}$$

(iii) relational algebra:

$$\pi_{drinker}(frequents \bowtie serves \bowtie likes)$$

(iv) SQL:

**select** *frequents.drinker*
**from** *frequents*, *serves*, *likes*
**where** *frequents.bar* = *serves.bar*
and *serves.beer* = *likes.beer*
and *likes.drinker* = *frequents.drinker*

(v) QBE:

| *frequents* | *drinker* | *bar* |
|---|---|---|
| | _d | _b |

| *serves* | *bar* | *beer* |
|---|---|---|
| | _b | _c |

| *likes* | *drinker* | *beer* |
|---|---|---|
| | _d | _c |

| *answer* | *drinker* |
|---|---|
| **I.** | _d |

(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:

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

(ii) domain calculus:

$$\{d : drinker \mid \forall b[frequents(d, b) \rightarrow$$
$$\exists c(serves(b, c) \land likes(d, c))]\}$$

(iii) relational algebra:

$$\pi_{drinker}(frequents) - \pi_{drinker}[frequents - \pi_{drinker,bar}(serves \bowtie likes)].$$

(iv) SQL:

```
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))
```

(v) QBE:

**Stage I** (find compatible drinker-bar pairs):

| *serves* | *bar* | *beer* |
|---|---|---|
| | _b | _c |

| *likes* | *drinker* | *beer* |
|---|---|---|
| | _d | _c |

| *compatible* | *drinker* | *bar* |
|---|---|---|
| **I.** | _d | _b |

**Stage II** (find drinkers who frequent some incompatible bar):

| *frequents* | *drinker* | *bar* |
|---|---|---|
| | _d | _b |

| *compatible* | *drinker* | *bar* |
|---|---|---|
| ¬ | _d | _b |

| *bad* | *drinker* |
|---|---|
| **I.** | _d |

**Stage III** (find answer by taking the complement):

| *frequents* | *drinker* | *bar* |
|---|---|---|
| | _d | |

| *bad* | *drinker* |
|---|---|
| ¬ | _d |

| *answer* | *drinker* |
|---|---|
| **I.** | _d |

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

This is just the complement of (b).

2. (a) List the actors cast in at least one movie by Kurosawa (here and below, $K$ stands for Kurosawa).

(i) domain calculus:

$$\{a:\ actor \mid \exists t\ film(t,K,a)\}.$$

(ii) relational algebra:

$$\pi_{actor}(\sigma_{director=K}(film)).$$

(iii) SQL:

**select** *actor*
**from** *film*
**where** *director* = *K*

(iv) QBE:

| film | title | director | actor |
|---|---|---|---|
| | | K | **P.** |

(b) List the actors cast in every movie by Kurosawa.

(i) domain calculus:

$$\{a:\ actor \mid \forall t[\exists x\ (film(t,K,x)) \rightarrow\ film(t,K,a)\}.$$

(ii) relational algebra:

$$\pi_{actor}(film) - \pi_{actor}[\pi_{title}(\sigma_{director=K}(film)) \bowtie \pi_{actor}(film) \\ -\pi_{title,actor}(\sigma_{director=K}(film))]$$

Also,

$$\pi_{title,actor}(film) \div \pi_{title}(\sigma_{director=K}(film)).$$

(iii) SQL:

> **select** *actor*
> **from** *film*
> **where** *actor* **not in**
> (**select** *f.actor*
> **from** *film f g*
> **where** *g.director* = *K* **and** *f.actor* **not in**
> (**select** *actor*
> **from** *film*
> **where** *title* = *g.title* ))

(iv) QBE:

**Stage I**

| film | title | director | actor |
|---|---|---|---|
| | | | _a |
| | _t | K | |
| ¬ | _t | | _a |

| bad-actors | actor |
|---|---|
| **I.** | _a |

**Stage II**

| film | title | director | actor |
|---|---|---|---|
| | | | _a |

| bad-actors | actor |
|---|---|
| ¬ | _a |

| answer | actor |
|---|---|
| **I.** | _a |

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

(i) domain calculus:

$$\{a : actor \mid \forall t \forall d[film(t,d,a) \rightarrow d = K]\}$$

(ii) relational algebra:

$$\pi_{actor}(film) - \pi_{actor}(\sigma_{director \neq K}(film)).$$

(iii) SQL:

**select** *actor*
**from** *film*
**where** *actor* **not in**
    (**select** *actor*
    **from** *film*
    **where** *director* $\neq$ $K$ )

(iv) QBE:

**Stage I**

| film | title | director | actor |
|---|---|---|---|
| | | $\neq$ K | _a |

| bad-actors | actor |
|---|---|
| **I.** | _a |

**Stage II**

| film | title | director | actor |
|---|---|---|---|
| | | | _a |

| bad-actors | actor |
|---|---|
| $\neg$ | _a |

| answer | actor |
|---|---|
| **I.** | _a |

(d) List all pairs of actors who act together in at least one movie.

(i) domain calculus:

$$\{a_1 : actor_1, a_2 : actor_2 \mid \exists t \exists d(film(t, d, a_1) \wedge film(t, d, a_2))\}$$

(ii) relational algebra:

$$\pi_{actor_1, actor_2}[\delta_{actor \to actor_1}(film) \bowtie \delta_{actor \to actor_2}(film)]$$

(iii) SQL:
**select** *f.actor, g.actor*
**from** *film f g*
**where** *f.title = g.title*

(iv) QBE:

| film | title | director | actor |
|---|---|---|---|
| | _t | | $\_a_1$ |
| | _t | | $\_a_2$ |

| answer | $\text{actor}_1$ | $\text{actor}_2$ |
|---|---|---|
| **I.** | $\_a_1$ | $\_a_2$ |

(e) List all pairs of actors cast in exactly the same movies.

(i) domain calculus:

$$\{a_1 : actor_1, a_1 : actor_2 \mid \forall t \forall d [film(t,d,a_1) \leftrightarrow film(t,d,a_2)]\}$$

(ii) relational algebra:
Consider first the query $q$ asking for pairs of actors $< a, a' >$ such that $a$ acts in every movie where $a'$ acts. This is expressed by:

$$\pi_{actor}(film) \times \delta_{actor \to actor'}(\pi_{actor}(film)) -$$
$$\pi_{actor,actor'}[\pi_{actor}(film) \times \delta_{actor \to actor'}(film) \; - \; film \times \delta_{actor \to actor'}(\pi_{actor}(film))]$$

The given query is now expressed by:

$$q \bowtie \delta_{actor,actor' \to actor',actor}(q).$$

(iii) SQL:

**select** *f.actor, g.actor*
**from** *film f g*
**where** <*f.actor, g.actor*> **not in** Q
**and** <*g.actor, f.actor*> **not in** Q

where Q is:

**select** *x.actor, y.actor*
**from** *film x y*
**where** *y.actor* **not in**
(**select** *actor*
**from** *film*
**where** *title* = *x.title*)

(iv) QBE:

**Stage I**

| film | title | director | actor |
|---|---|---|---|
| | | | _a$_1$ |
| | _t | | _a$_2$ |
| ¬ | _t | | _a$_1$ |

| bad | actor$_1$ | actor$_2$ |
|---|---|---|
| **I.** | _a$_1$ | _a$_2$ |

**Stage II**

| film | title | director | actor |
|---|---|---|---|
| | | | _a$_1$ |
| | | | _a$_2$ |

| bad | actor$_1$ | actor$_2$ |
|---|---|---|
| ¬ | _a$_1$ | _a$_2$ |
| ¬ | _a$_2$ | _a$_1$ |

| answer | actor$_1$ | actor$_2$ |
|---|---|---|
| **I.** | _a$_1$ | _a$_2$ |

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

(i) domain calculus:

$$\{d:\ director \mid \forall a[\exists t \exists e(film(t,e,a)) \rightarrow \exists t film(t,d,a)]\}$$

(ii) relational algebra:

$$\pi_{director,actor}(film) \div \pi_{actor}(film).$$

(iii) SQL:

**select** *director*
**from** *film*
**where** *director* **not in**
    (**select** *f.director*
    **from** *film f g*
    **where** *f.director* **not in**
        (**select** *director*
        **from** *film*
        **where** *actor* = *g.actor* ))

(iv) QBE:

**Stage I**

| film | title | director | actor |
|---|---|---|---|
| | | _d | |
| | | | _a |
| ¬ | | _d | _a |

| bad-directors | director |
|---|---|
| **I.** | _d |

**Stage II**

| film | title | director | actor |
|---|---|---|---|
| | | _d | |

| bad-directors | director |
|---|---|
| ¬ | _d |

| answer | director |
|---|---|
| **I.** | _d |

3. (a) True. The proof uses distributivity of existential quantification over disjunction.

   (b) False (note that this says, essentially, that existential quantifica tion distributes over conjunction). A counterexample is:

| $R$ | $A$ | $B$ |
|---|---|---|
| | 0 | 0 |

| $S$ | $A$ | $B$ |
|---|---|---|
| | 0 | 1 |

   (c) False. Same counterxample as for (b).

4. One way to make the evaluation more efficient is to rewrite the query as

$$\pi_{director}[\pi_{title,director}(movie) \bowtie \pi_{title}(\sigma_{theater \neq "Cove"}(schedule))]$$

This cuts down the size of relations participating in the join and avoids building up an unnecessarily large intermediate result (resulting from the join).

5. Consider the following rspj algebra query ($R$ is a relation with attributes $ABC$):

$$\pi_{AC}[\pi_{AB}(R) \bowtie \pi_{BC}(R)] \bowtie \pi_A[\pi_{AC}(R) \bowtie \pi_{CB}(R)].$$

   (a) The tableau corresponding to the query is:

| $A$ | $B$ | $C$ |
|---|---|---|
| $a$ | $b_1$ | $c_1$ |
| $a_1$ | $b_1$ | $c$ |
| $a$ | $b_2$ | $c_2$ |
| $a_2$ | $b_3$ | $c_2$ |
| $a$ | | $c$ |

(b) There is a homomorphism of the tableau mapping the 2nd and 3rd rows to the first. So the minimized tableau is:

| $A$ | $B$ | $C$ |
|---|---|---|
| $a$ | $b_1$ | $c_1$ |
| $a_1$ | $b_1$ | $c$ |
| $a$ | | $c$ |

(c) An rspj algebra query corresponding to the minimized tableau is:

$$\pi_{AC}(\pi_{AB}(R) \bowtie \pi_{BC}(R)).$$

(d) Redo (a),(b),(c) for the query:

$$\pi_{AC}[\pi_{AB}(R) \bowtie \pi_{BC}(\sigma_{B=8}(R))] \bowtie \pi_A[\pi_{AC}(\sigma_{A=5}(R)) \bowtie \pi_{CB}(R)].$$

The corresponding tableau is:

| $A$ | $B$ | $C$ |
|---|---|---|
| 5 | 8 | $c_1$ |
| $a_1$ | 8 | $c$ |
| 5 | $b_1$ | $c_2$ |
| $a_2$ | $b_2$ | $c_2$ |
| 5 | | $c$ |

The last two rows can be mapped to the first. The minimized tableau is:

| $A$ | $B$ | $C$ |
|---|---|---|
| 5 | 8 | $c_1$ |
| $a_1$ | 8 | $c$ |
| 5 | | $c$ |

A corresponding algebra query is:

$$\pi_{AC}(\sigma_{A=5}\sigma_{B=8}(R)) \bowtie \pi_C(\sigma_{B=8}(R)).$$

6. (i) The tableau $T$ corresponding to $q$ is

$$\begin{array}{ccc} A & B & C \\ \hline a & b_1 & c_1 \\ a_1 & b & c_1 \\ a_2 & b & c_2 \\ a_2 & b_2 & c \\ \hline a & b & c \end{array}$$

Chasing with respect to $\Sigma = \{AC \to B, B \to C, C \to A\}$ yields:

$$CHASE_\Sigma(T) = \begin{array}{ccc} A & B & C \\ \hline a & b & c_1 \\ a & b & c_1 \\ a & b & c_1 \\ a & b_2 & c \\ \hline a & b & c \end{array}$$

Eliminating duplicate rows from $CHASE_\Sigma(T)$ yields the minimal tableau:

$$\begin{array}{ccc} A & B & C \\ \hline a & b & c_1 \\ a & b_2 & c \\ \hline a & b & c \end{array}$$

and the corresponding rspj query is

$$\pi_{AB}(R) \bowtie \pi_{AC}(R).$$

(ii)The tableau $T$ corresponding to $q$ is

$$\begin{array}{ccc} A & B & C \\ \hline a & b_1 & 5 \\ a_1 & b & 5 \\ 8 & b & c_2 \\ 8 & b_2 & c \\ \hline a & b & c \end{array}$$

Chasing with respect to $\Sigma = \{AC \to B, B \to C, C \to A\}$ yields (after eliminating duplicate rows):

$$CHASE_\Sigma(T) = \begin{array}{ccc} A & B & C \\ \hline 8 & b & 5 \\ 8 & b_2 & c \\ \hline a & b & c \end{array}$$

This tablueau is minimal, and the corresponding rspj query is

$$\pi_{AB}(\sigma_{A=8}(\sigma_{C=5}(R))) \bowtie \pi_{AC}(\sigma_{A=8}(R)).$$