Practice Problems RA & SQL

Introduction to Databases

CompSci 316 Spring 2019

DUKE COMPUTER SCIENCE

Announcements (Tue., Feb. 5)

- Homework #1 can be submitted (Feb 8) 11:59pm
  - HW2 SQL problems to be released today
  - Will be due in 2 weeks and SQL is included in midterm
- Please fill out the polls on piazza
  - Thanks if you already did!
- Milestone 1 for project due on Feb 26 (Tuesday) in 3 weeks
- Midterm in class in two weeks Feb 19 (Tuesday)
  - Everything covered until the class before 2/14 is included

Problem 1:

- Find names of all drinkers who frequent only those bars that serve some beers they like.

- Drinker(name, address)
- Bar(name, address)
- Beer(name, brewer)
- Frequents(drinker, bar, times)
- Likes(drinker, beer)
- Serves(bar, beer, price)

Problem 1:

Q: $L \bowtie S$

$\Pi\ L \bowtie F \bowtie S$

- Find names of all drinkers who frequent only those bars that serve some beers they like.

$\Pi_{name}\ Drinker = [$

- Drinker(name, address)
- Bar(name, address)
- Beer(name, brewer)
- Frequents(drinker, bar, times)
- Likes(drinker, beer)
- Serves(bar, beer, price)

$\Pi_{drinker} [\Pi\ frequents$

$- \ \Pi_{drinker,beer}\ L \bowtie F \bowtie S$

$(\Pi_{drinker,bar} (likes \bowtie_{beer=beer1} \rho_{bar,beer1,price}\ Serves))]$

RA Query

- // general idea: (drinkers who frequent only those bars that serve some beers they like) =
- // (drinkers) - (drinkers who frequent some bar but like none of the beers served there).
- // first, let us find all (drinker, bar) pairs where the bar serves some beer that the drinker likes:
- e1 :- \project_{drinker, bar} (likes \join serves);
- // then, we find all drinkers who frequent some bar that does not serve any beer they like:
- e2 :- \project_{drinker} (\project_{drinker, bar} frequents \diff e1);
- // finally, the answer is given by:
- \project_{name} drinker \diff e2;

SQL Query

- Find names of all drinkers who frequent only those bars that serve some beers they like.

- Drinker(name, address)
- Bar(name, address)
- Beer(name, brewer)
- Frequents(drinker, bar, times)
- Likes(drinker, beer)
- Serves(bar, beer, price)

SQL Query

```
SELECT name
FROM drinker
WHERE NOT EXISTS
        (SELECT bar -- frequented by drinker but
                    -- not serving beers liked by drinker
        FROM frequents
        WHERE drinker = name
                AND bar NOT IN
                        (SELECT bar
                         FROM serves, likes
                         WHERE drinker = name
                               AND serves.beer = likes.beer) );
```

Problem-2

- Find all (bar1, bar2) pairs where the set of beer served at bar1 is a proper subset of those served at bar2;
- i.e., bar2 serves every beer that bar1 serves and plus some more.

- Drinker(name, address)
- Bar(name, address)
- Beer(name, brewer)
- Frequents(drinker, bar, times)
- Likes(drinker, beer)
- Serves(bar, beer, price)

Problem-2

b1, r1
b1, r2
b2, r1
b2, r2

- Find all (bar1, bar2) pairs where the set of beer served at bar1 is a proper subset of those served at bar2;
- i.e., bar2 serves every beer that bar1 serves and plus some more.

1 bar, beer

e1 = $[\Pi_{bar,beer}(Beer \times Bar) - Serves]$ e2

- Drinker(name, address)
- Bar(name, address)
- Beer(name, brewer)
- Frequents(drinker, bar, times)
- Likes(drinker, beer)
- Serves(bar, beer, price)

e1 $\bowtie_{beer=beer}$ Serves

2 bar1, bar2, beer

3 $(bar \times bar) - e2$

$\Pi_{bar2,bar1} e2 - \Pi_{bar1,bar2} e2$

RA Query – soln (a)

- // compute does-not-serve info below in e1
- e1 :- (\rename_{bar} \project_{name} Bar \cross \rename_{beer} \project_{name} Beer) \diff \project_{bar, beer} Serves);
- // in e2 below, bar1 serves beer, while bar2 does not:
- e2 :- \rename_{bar1, beer, bar2} ((\project_{bar, beer} Serves) \join_{bar1 <> bar2 and beer = beer2} \rename_{bar2, beer2} e1);
- // so, a pair (bar1, bar2) appears in e2 <=> it's NOT the case that bar2 serves every beer that bar1 serves. therefore, an answer pair (a,b) must not appear in e2 as (bar1, bar2), but must appear in e2 as (bar2, bar1) to ensure that b serves some beer that a doesn't:
- \project_{bar2, bar1} e2 \diff \project_{bar1, bar2} e2;

RA Query – soln (b)

- // in e1 below, bar1 serves beer, and is paired with every bar2:
- e1 :- \rename_{bar1, beer, bar2} (\project_{bar, beer} serves \cross \project_{name} bar);
- // in e2 below, bar1 serves beer, while bar2 does not:
- e2 :- \rename_{bar1, beer, bar2} (e1 \diff \project_{bar2, beer, bar1} e1);
- // so, a pair (bar1, bar2) appears in e2 <=> it's NOT the case that bar2 serves every beer that bar1 serves. therefore, an answer pair (a,b) must not appear in e2 as (bar1, bar2), but must appear in e2 as (bar2, bar1) to ensure that b serves some beer that a doesn't:
- \project_{bar2, bar1} e2 \diff \project_{bar1, bar2} e2;

b1, r1
b1, r2
b2, r1

~~b1 r1 b1~~
~~b1 r1 b2~~
~~b1 r2 b1~~
b1 r2 b2
b2 r1 b1

~~b2 r1 b2~~

~~b1 r1 b1~~
b2 r1 b1
b1 r2 b1
b2 r2 b1
~~b1 r1 b2~~
b2 r1 b2

SQL Query

```
SELECT b1.name, b2.name
FROM bar b1, bar b2
WHERE NOT EXISTS  --- make sure that beers served at b2 is a subset of those at b1
            ((SELECT beer
              FROM serves
              WHERE bar = b1.name)
            EXCEPT
              (SELECT beer
               FROM serves
               WHERE bar = b2.name))
  AND (SELECT COUNT(*) --- make sure that the subset is proper
          FROM serves
          WHERE bar = b1.name)
          <
         (SELECT COUNT(*)
          FROM serves
          WHERE bar = b2.name);
```