CSC343 Worksheet 4 Solution

June 18, 2020

1. a) $[(1,0,1),(5,4,9),(1,0,1),(6,4,16),(7,9,16)]$

   b) $[(1,0),(3,3),(3,4),(4,3),(1,1),(4,3)]$

   c) $[(0,1),(0,1),(2,3),(2,4),(3,4)]$

   Notes:

   - $\tau_L(R)$ sorts tuples in order indicated by L.
     - e.g.

       $\tau_{C,B}(R)$ in $R(A,B,C)$ orders the tuples of R by their values of C, and tuples with the same C-value are ordered by their B value.

   d) $[(0,1),(0,2),(2,4),(2,5),(3,4),(3,4)]$

   e) $[(0,1),(2,4),(2,5),(3,4),(0,2)]$

   Notes:

   - $\delta(R)$ converts a bag into a set
     - e.g.

       Let $R = [(1,2),(3,4),(1,2),(1,2)]$

       $\delta(R(A,B)) = [(1,2),(3,4)]$

   f) $[(0,2),(2,7),(3,4)]$

   Notes:

   - $\gamma_L(R)$ is an operator that groups a relation and/or aggregate some columns.
     - L in $\gamma_L(R)$ is either
       1. **Grouping attribute** or an attribute by which R will be grouped.

2. **Aggregated attribute** or an attribute where an aggregation operator is applied to.

Example:

$\gamma_{starName,MIN(year)\rightarrow minYear,COUNT(title)\rightarrow ctTitle}(\text{StarsIn})$

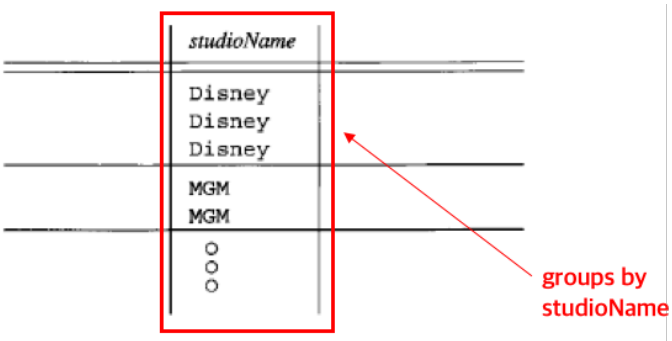

Figure 5.4: A relation with imaginary division into groups

g) $[(0, 1.5), (2, 4.5), (3, 4)]$

h) $[(0, 1), (0, 1), (2, 3), (2, 4), (3, 4)]$

i) $\gamma_{A,MAX(C)}([(2, 3, 4), (2, 3, 4)]) \rightarrow [(2, 4)]$

j) $[(0, 1, \perp), (2, 3, 4), (2, 3, 4), (0, 1, \perp), (2, 4, \perp), (3, 4, \perp)]$

Notes:

- $\overset{\circ}{\bowtie}$ is an outerjoin operator
  - $\overset{\circ}{\bowtie}_L$ means Natural Left Outer Join
  - $\overset{\circ}{\bowtie}_R$ means Natural Right Outer Join
  - $\overset{\circ}{\bowtie}$ means Natural Full Outer Join
  - $\perp$ means null
- e.g. $U \overset{\circ}{\bowtie} V$

A	B	C
1	2	3
4	5	6
7	8	9

(a) Relation U

B	C	D
2	3	10
2	3	11
6	7	12

(b) Relation V

A	B	C	D
1	2	3	10
1	2	3	11
4	5	6	$\perp$
7	8	9	$\perp$
$\perp$	6	7	12

(c) Result $U \overset{\circ}{\bowtie} V$

k) $[(\perp, 0, 1), (\perp, 2, 4), (\perp, 2, 5), (2, 3, 4), (\perp, 0, 2), (2, 3, 4)]$

l) $[(0, 1, \perp), (2, 3, 4), (2, 3, 4), (0, 1, \perp), (2, 4, \perp), (3, 4, \perp),$
$(\perp, 0, 1), (\perp, 2, 4), (\perp, 2, 5), (2, 3, 4), (\perp, 0, 2), (2, 3, 4)]$

m) $(0, 1) : \{(2, 4), (2, 5), (3, 4), (3, 4)\}$

But, $\{(2, 3), (2, 4), (3, 4)\}$ from R and $\{(0, 1), (0, 2)\}$ in S dont match. So,

$[(0, 1, 2, 4), (0, 1, 2, 5), (0, 1, 3, 4), (0, 1, 3, 4), (0, 1, 2, 4), (0, 1, 2, 5), (0, 1, 3, 4), (0, 1, 3, 4),$
$(2, 3, \perp, \perp), (2, 4, \perp, \perp), (3, 4, \perp, \perp), (\perp, \perp, 0, 1), (\perp, \perp, 0, 2)]$

Notes:

- $R \overset{\circ}{\bowtie}_C S$ is equivalent form of $\sigma_C(R \times S)$ but instead of filtering, the unmatching tuples filled with null.

2. a) SELECT model FROM PC WHERE speed ¿ 3.0;

b) SELECT DISTINCT maker FROM Products NATURAL JOIN Laptops WHERE hd >= 100;

c)
```
SELECT model, price FROM (
    (SELECT model, price FROM PC NATURAL JOIN Products)
    UNION
    (SELECT model, price FROM Laptop NATURAL JOIN Products)
    UNION
    (SELECT model, price FROM Printer INNER JOIN Products ON
Printer.model = Product.model)
);

```

d) SELECT model FROM Printer WHERE color;

e)
```
(SELECT DISTINCT makers FROM Products WHERE type='laptops') -
(SELECT DISTINCT makers FROM Products WHERE type='pc');
```

f)
```
SELECT hd FROM PC WHERE EXISTS (
    SELECT hd, COUNT(model) FROM PC GROUP BY hd
    HAVING COUNT(model) > 2
);
```

3. a) SELECT class, country FROM classes WHERE bore >= 16;

b) SELECT * FROM Ships WHERE launched < 1921;

c) SELECT * FROM Outcomes WHERE result='sunk';

d) SELECT * FROM Classes NATURAL JOIN Ships WHERE displacement > 35000;

e)
```
SELECT name, displacement, numGuns FROM Classes NATURAL JOIN (
    SELECT * FROM Ships INNER JOIN Outcomes ON Ships.name =
Outcome.ship
);
```

f)
```
(SELECT name FROM Ships)
UNION
(SELECT ship AS name FROM Outcomes);
```

g)
```
SELECT class, COUNT(class) FROM Ships
GROUP BY Class
HAVING COUNT(class) = 1;
```

h)
```
(SELECT countries FROM Classes WHERE type='bb')
INTERSECT
(SELECT countries FROM Classes WHERE type='bc');
```

i) Current attempt:
```
(SELECT Table1.name FROM Outcomes AS Table1 INNER JOIN Ships ON
Outcomes.ship = Ships.name)
```

Took too much time. Omitted for now.

4. a) SELECT AVG(speed) FROM PC;

b) SELECT AVG(speed) FROM Laptop HAVING price > 1000;

c) SELECT AVG(price) FROM PC NATURAL JOIN Product HAVING maker = 'A';

d)
```
SELECT AVG(price) FROM (
    (SELECT model, price FROM PC NATURAL JOIN Product WHERE maker
= 'D')
    UNION
    (SELECT model, price FROM Laptop NATURAL JOIN Product WHERE
maker = 'D')
);

```

e) SELECT speed, AVG(price) FROM PC GROUP BY speed;

f) SELECT maker, AVG(screen) FROM Laptop NATURAL JOIN Product GROUP BY maker;

g)
```
SELECT maker, COUNT(model) FROM Products GROUP BY maker HAVING
COUNT(model) >= 3;

```

h)
```
SELECT maker, MAX(price) FROM PC
NATURAL JOIN Products GROUP BY maker;

```

i)
```
SELECT speed, AVG(price) FROM PC GROUP BY speed
HAVING speed > 2.0;

```

5. a) SELECT COUNT(class) FROM Classes;

Correct Solution:

SELECT COUNT(class) FROM Classes HAVING type='bb';

b) SELECT class, AVG(numGuns) FROM Classes GROUP BY class;

Correct Solution:

SELECT class, AVG(numGuns) FROM Classes GROUP BY class
HAVING type='bb';

c) SELECT AVG(numGuns) FROM Classes HAVING type='bb';

d) SELECT class, MIN(launched) FROM Ships GROUP BY class;

e)
```
SELECT class, COUNT(Ships.name) FROM Ships INNER JOIN Outcomes
GROUP BY class ON Ships.name = Outcomes.ship
HAVING result='sunk';

```

6.
```
SELECT name, MIN(movieYear) FROM MovieStar INNER JOIN StarsIn ON
StarsIn.starName = MovieStar.name GROUP BY name HAVING COUNT(name)
>= 3;

```

7. Yes. It is possible

   - Rename aggregate columns using ρ
   - Use σ around $\rho(\gamma(\cdots))$

8. a) INSERT INTO PC(model, speed, ram, hd, price) VALUES (1100, 3.2, 1024, 180, 2499);

   INSERT INTO Product(maker, model, type) VALUES ('C', 1100, 'pc');

   Notes:

   - Insertion
     - **Syntax:** INSERT INTO R($A_1, A_2, ..., A_n$) VALUES ($v_1, ..., v_n$)
     - Inserts a tuple with values into relation R
     - Values can be from select statement

```
\\ Example 1
INSERT INTO StarsIn(movieTitle, movieYear, starName)
VALUES('The Maltese Fiction', 1942, 'Sydney Greenstreet');

\\ Example 2
INSERT INTO Studio(name)
    SELECT DISTINCT studioName
    FROM Movies
    WHERE studioName NOT IN
        (SELECT name FROM Studio);
```

b)

```
// SQL statement 1
INSERT INTO Laptop(model, speed, ram, hd, screen, price)
    SELECT DISTINCT model + 110, speed, ram, hd, 17 AS screen,
price + 500
    FROM PC;

// SQL statement 2
INSERT INTO Product(maker, model, type)
    SELECT DISTINCT maker, model + 110, 'laptop' AS type
    FROM Product WHERE type='pc';
```

c) **Notes:**

   - Deletion
     - **Syntax:** DELETE FROM R WHERE < Condition >;

```
DELETE FROM StarsIn
WHERE movieTitle = 'The Maltest Falcon' AND
      movieYeear = 1942 AND
      starName = 'Sydney Greenstreet';
```