

CSC343 Worksheet 3 Solution

June 17, 2020

1. Exercise 6.1.1:

If there is a comma between A and B (i.e, *SELECT A, B*), we can conclude A and B are two different attributes.

If there are no commas between A and B , we can conclude B is an alias of A .

2. Exercise 6.1.2:

- a) *SELECT address FROM Studio WHERE name = 'MGM';*
- b) *SELECT birthdate FROM MovieStar WHERE name = 'Sandra Bullock';*
- c) *SELECT starName FROM StarsIn WHERE movieYear = 1980, movieTitle LIKE '%Love%';*

Correct Solution:

```
SELECT starName FROM StarsIn WHERE movieYear = 1980 AND movieTitle  
LIKE '%Love%';
```

- d) *SELECT name FROM MovieExec WHERE netWorth >= 10000000;*
- e) *SELECT name FROM MovieStar WHERE gender='male' OR address LIKE '%Malibu%';*

3. Exercise 6.1.3:

- a) *SELECT model, speed, hd FROM PC WHERE price < 1000;*
- b) *SELECT model, speed AS gigahertz, hd AS gigabytes FROM PC WHERE price < 1000;*
- c) *SELECT maker FROM Product WHERE type='printer';*
- d) *SELECT model, ram, screen FROM Laptops WHERE price > 1500;*
- e) *SELECT * FROM Printer WHERE color=TRUE;*

f) `SELECT model, hd FROM PC WHERE speed = 3.20 AND price < 2000;`

4. **Exercise 6.1.4:**

- a) `SELECT class, country FROM Classes where numGuns >= 10;`
- b) `SELECT name AS shipName FROM Ships WHERE launched < 1918;`
- c) `SELECT ship, battle FROM Outcomes WHERE result='sunk';`
- d) `SELECT name FROM Ships WHERE name = class;`
- e) `SELECT name FROM Ships WHERE name LIKE 'R%';`
- f) `SELECT name FROM ships WHERE name LIKE '% % %';`

5. **Exercise 6.1.5:**

- a) Given $a = 10$, the sets of tuples that satisfy the condition is

$(10, -MAX_INT), (10, -MAX_INT + 1), \dots, (10, 0), \dots, (10, MAX_INT - 1),$
 $(10, MAX_INT), (10, NULL)$

Given $b = 20$, the sets of tuples that satisfy the condition is

$(-MAX_INT, 20), (-MAX_INT + 1, 20), \dots, (0, 20), \dots, (MAX_INT - 1, 20),$
 $(MAX_INT, 20), (NULL, 20)$

Given $a = 10$ and $b = 20$, the set of tuple that satisfy the condition is $(10, 20)$

- b) Given $a = 10$ AND $b = 20$, the only set of (a, b) tuple that satisfy the condition is $(10, 20)$.
- c) There are three cases to consider
 - i. $a < 10$

In this case, the set of (a, b) tuples that satisfy the condition is:

$(9, -MAX_INT), (9, -MAX_INT + 1), \dots, (9, 0), \dots, (9, MAX_INT - 1),$
 $(9, MAX_INT), (9, NULL)$

$(8, -MAX_INT), (8, -MAX_INT + 1), \dots, (8, 0), \dots, (8, MAX_INT - 1),$
 $(8, MAX_INT), (8, NULL)$

...

$(-MAX_INT + 1, -MAX_INT), (-MAX_INT + 1, -MAX_INT + 1),$
 $\dots, (-MAX_INT + 1, 0), \dots, (-MAX_INT + 1, MAX_INT - 1),$
 $(-MAX_INT + 1, MAX_INT), (-MAX_INT + 1, NULL)$

$(-MAX_INT + 1, -MAX_INT), (-MAX_INT + 1, -MAX_INT + 1),$
 $\dots, (-MAX_INT + 1, 0), \dots, (-MAX_INT + 1, MAX_INT - 1),$
 $(-MAX_INT + 1, MAX_INT), (-MAX_INT + 1, NULL)$

ii. $a \geq 10$

In this case, the set of (a, b) tuples that satisfy the condition is:

$(10, -MAX_INT), (10, -MAX_INT + 1), \dots, (10, 0), \dots, (10, MAX_INT - 1),$
 $(10, MAX_INT), (10, NULL)$

$(11, -MAX_INT), (11, -MAX_INT + 1), \dots, (11, 0), \dots, (11, MAX_INT - 1),$
 $(11, MAX_INT), (11, NULL)$

...

$(MAX_INT - 1, -MAX_INT), (MAX_INT - 1, -MAX_INT + 1),$
 $\dots, (MAX_INT - 1, 0), \dots, (MAX_INT - 1, MAX_INT - 1),$
 $(MAX_INT - 1, MAX_INT), (MAX_INT - 1, NULL)$

$(MAX_INT, -MAX_INT), (MAX_INT, -MAX_INT + 1),$
 $\dots, (MAX_INT, 0), \dots, (MAX_INT, MAX_INT - 1),$
 $(MAX_INT, MAX_INT), (MAX_INT, NULL)$

iii. $a < 10$ AND $a \geq 10$

This case is not considered. No (a, b) tuples match this condition.

d) In this case the set of (a, b) tuples that satisfy this condition is

$(-MAX_INT, -MAX_INT), (-MAX_INT + 1, -MAX_INT + 1),$
 $\dots, (0, 0), \dots, (MAX_INT - 1, MAX_INT - 1),$
 (MAX_INT, MAX_INT)

Here, the case $a = NULL$ and $b = NULL$ is not considered, since $NULL \neq NULL$.

Notes:

- $NULL = NULL$ is $NULL$.

e) In this case, the set of (a, b) tuples that satisfy this condition is

$(-MAX_INT, -MAX_INT), (-MAX_INT, -MAX_INT + 1),$
 $\dots, (-MAX_INT, MAX_INT - 1),$
 $(-MAX_INT, MAX_INT),$

$(-MAX_INT + 1, -MAX_INT + 1), (-MAX_INT + 1, -MAX_INT + 2),$
 $\dots, (-MAX_INT + 1, MAX_INT - 1),$
 $(-MAX_INT + 1, MAX_INT),$

...

$(MAX_INT - 1, MAX_INT - 1), (MAX_INT - 1, MAX_INT),$
 (MAX_INT, MAX_INT)

Here, the case $a = NULL$ OR $b = NULL$ is not considered, since $a \not\leq b$.

6. SELECT * FROM Movies WHERE length;
7. (a) SELECT StarsIn.starName FROM StarsIn, MovieStar WHERE
StarsIn.starName = MovieStar.name AND MovieStar.gender = 'male';
 (b) SELECT StarsIn.starName FROM Movies, StarsIn WHERE
StarsIn.movieTitle = Movies.title AND Movies.studioName = 'MGM';
 (c) SELECT MovieExec.name FROM MovieExec, Studio WHERE MovieExec cert# =
studio.presC# AND Studio.name = 'MGM';
 (d) SELECT M2.title FROM Movies AS M1, Movies AS M2 WHERE
M1.title = "Gone With the Wind" AND M2.length > M1.length;
 (e) SELECT Mx2.name FROM MovieExec AS Mx1, MovieExec AS Mx2 WHERE
Mx1.name = 'Merg Griffin' AND Mx2.netWorth > Mx1.netWorth;
8. a) SELECT Product.maker, Laptop.speed FROM Product, Laptops WHERE
Product.type = 'laptop' AND Laptop.hd >= 30;
 b) (SELECT model, price FROM PC INNER JOIN Product ON
PC.model = Product.model WHERE maker = 'B')

UNION

(SELECT model, price FROM Printer INNER JOIN Product ON
Printer.model = Product.model WHERE maker = 'B')

UNION

(SELECT model, price FROM Laptop INNER JOIN Product ON
Laptop.model = Product.model WHERE maker = 'B')

- c) (SELECT maker FROM Product WHERE type='laptop') -
(SELECT maker FROM Product WHERE type='pc')
- d) SELECT pc1.hd FROM PC AS pc1, PC AS pc2 WHERE
pc1.model != pc2.model AND pc1.hd = pc2.hd;
- e) SELECT pc1.model FROM PC AS pc1, PC AS pc2 WHERE
pc2.model != pc1.model AND
pc2.model >= pc1.model AND
pc2.ram = pc1.ram AND
pc2.speed = pc1.speed;

9. The second part of problem (i.e. Writing each query in different ways) will be done during review :).

```
a)  SELECT maker FROM Product WHERE model IN (
      SELECT model FROM PC WHERE product.model = PC.model AND
      PC.speed >= 3.0
    );
```

```
b)  SELECT p1.model FROM Printer AS p1 WHERE
      p1.price >= ALL (
      SELECT p2.price FROM Printer AS p2
    )
```

```
c)  SELECT l1.model FROM Laptop AS l1 WHERE
      speed >= ALL (
      SELECT l2.speed FROM Laptop AS l2
    )
```

Correct Solution:

```
1  SELECT l1.model FROM Laptop AS l1 WHERE
2  speed <= ALL (                                //correction: >=
   changed to <=
3  SELECT l2.speed FROM Laptop AS l2
4  )
5
```

```
d)  SELECT model FROM (
      (SELECT model, price FROM PC)
      UNION
      (SELECT model, price FROM Laptop)
      UNION
      (SELECT model, price FROM Printer)
    ) AS ModelPrice WHERE price >= ANY (
      SELECT price FROM ModelPrice
    )
```

```
e)  SELECT model FROM (
      (SELECT model, price FROM PC)
      UNION
      (SELECT model, price FROM Laptop)
      UNION
      (SELECT model, price FROM Printer)
    ) AS ModelPrice WHERE price >= ANY (
      SELECT price FROM ModelPrice
    )
```

```

f)  SELECT maker FROM Product, Printer WHERE
    Product.model = Printer.model AND
    Printer.color = TRUE AND
    Printer.price <= ANY (
        SELECT price FROM Printer
    );

```

Notes:

- EXISTS

- EXISTS R is a condition that is true if and only if relation R is not empty

```

1  SELECT SupplierName
2  FROM Suppliers
3  WHERE EXISTS (SELECT ProductName FROM Products WHERE
4  Products.SupplierID = Suppliers.supplierID AND Price = 22);

```

- s IN R

- is true if and only if s is equal to one of the values in R .
 - s NOT IN R true if and only if s has no value in R .

```

1  SELECT name
2  FROM MovieExec
3  WHERE cert# IN
4      (SELECT producerC#
5       FROM Movies
6       WHERE (title, year) IN
7           (SELECT movieTitle movieYear
8            FROM StarsIn
9            WHERE starName = 'Harrison Ford'
10          )
11      );
12

```

- $s > \text{ANY } R$

- is true if and only if s is greater than at least one value in unary relation R .

- $s > \text{ALL } R$

- is true if and only if s is greater than at least one value in unary relation R .

```

10. a) SELECT country FROM Classes WHERE
    numGuns >= ANY (
        SELECT numGuns FROM Classes
    );

```

```

b)  SELECT name FROM Ships WHERE EXISTS (
      SELECT * FROM Outcome WHERE
      Ships.name = Outcomes.ship AND
      Outcome.result = 'sunk'
    );

```

```

c)  SELECT name FROM Ships WHERE EXISTS (
      SELECT name FROM Ships, Classes WHERE
      Ships.class = Classes.class AND
      Classes.bore = 16
    );

```

```

d)  SELECT battle FROM Outcomes WHERE EXISTS (
      WHERE EXISTS (
      SELECT * FROM Ships WHERE
      Outcomes.ship = Ships.name AND
      Ships.class = 'Kongo'
    )
    );

```

11.

12.

13. a) Cross join would result in the following attributes

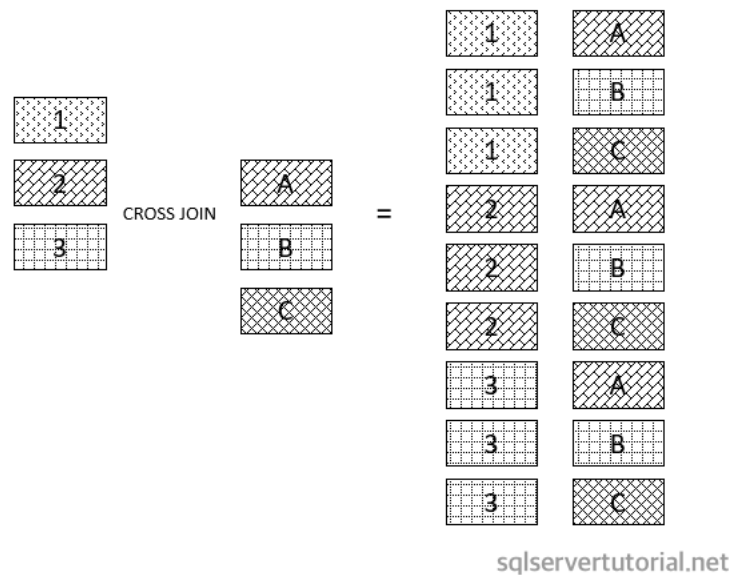
(Studio.name, Studio.address, Studio.pressC#, MovieExec.name,
MovieExec.address, MovieExec.cert#, MovieExec.networth)

With its tuples containing all possible combinations of values

Notes:

- **Cross Join:**

- Is equivalent form of $R \times S$
- Creates all possible combinations of values while keeping all all columns.



b) In this case, the resulting operation would have 7 attributes

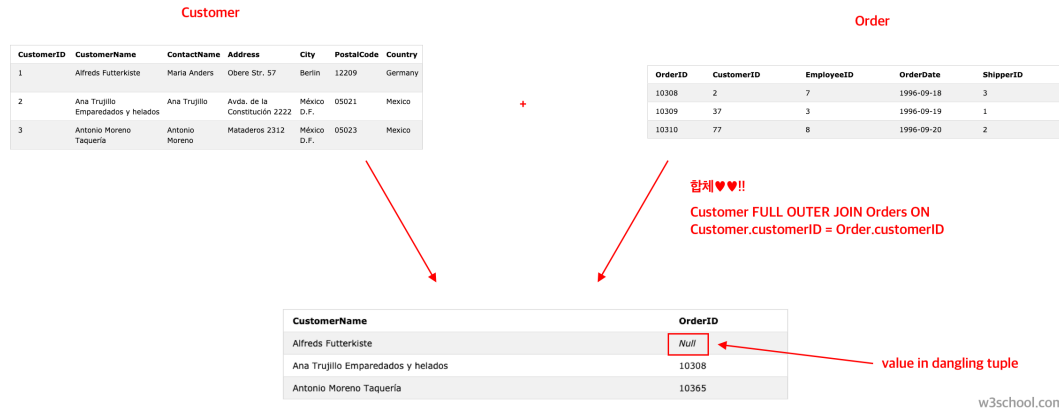
(StarsIn.movieTitle, StarsIn.moveYear, StarsIn.starName,
MovieStar.name, MovieStar.addresss, MovieStar.gender,
MovieStar.birthDate)

with attribute values of MovieStar returning null when StarsIn values are present, and vice versa

Notes:

- **Outerjoins:**

- Is equivalent form of Natural Join but with missing values in rows (i.e. dangling tuples) returning null
- **Syntax:** *Relation 1* NATURAL FULL OUTER JOIN *Relation 2*
- FULL OUTER JOIN VS NATURAL FULL OUTER JOIN
 - * FULL OUTER JOIN
 - Allows to explicitly define the keys for the join condition
 - * NATURAL FULL OUTER JOIN
 - Database engine chooses the keys based on common names



c) In this case, the resulting operation would have 6 attributes

(StarsIn.movieTitle, StarsIn.moveYear, MovieStar.name, MovieStar.addresss, MovieStar.gender, MovieStar.birthDate)

and the attribute values of MovieStar not present in StarsIn are padded with null, and vice versa

```

14 SELECT * FROM
15 (SELECT * FROM PC NATURAL LEFT OUTER JOIN Product)
16 NATURAL FULL OUTER JOIN
17 (SELECT * FROM Laptop NATURAL LEFT OUTER JOIN Product)
18 NATURAL FULL OUTER JOIN
19 (SELECT p.model, p.color, p.type AS printType, p.price, pr.maker, pr
20 .type FROM
21 Printer AS p LEFT FULL OUTER JOIN
22 Product AS pr ON Printer.model = Product.model)
23

```

Correct Solution:

```

1 SELECT * FROM
2 (SELECT * FROM PC NATURAL LEFT OUTER JOIN Product)
3 NATURAL FULL OUTER JOIN
4 (SELECT * FROM Laptop NATURAL LEFT OUTER JOIN Product)
5 NATURAL FULL OUTER JOIN
6 (SELECT p.model, p.color, p.type AS printType, p.price, pr.maker,
7 pr.type FROM
8 Printer AS p LEFT OUTER JOIN // Corrected
9 Product AS pr ON Printer.model = Product.model)

```

```

15 SELECT * FROM Ships NATURAL LEFT OUTER JOIN Classes;
16

```

```

16 SELECT * FROM Ships NATURAL FULL OUTER JOIN Classes
17 WHERE ship = class;
18

```