Noureldeen Allama

900202115

DB Assignment 4

QUESTION (1):

(A):

Computer Store Database

RA Expression:

1- π (manufacturer, model) (σ (price >= 500 AND price <= 800) (PC))

2- π (model) (PC - (σ (PC.manufacturer = Laptop.manufacturer) (PC ⋈ Laptop)))

3- π (Laptop.model, Laptop.speed, Laptop.price) (Laptop ⋈ Printer (manufacturer = Laptop.manufacturer AND is_colored = TRUE))

4- π (manufacturer) (PC) ÷ (π (speed) (PC))

RC Expression:

1- {p | ∃pce ∈ PC: p.manufacturer = pce.manufacturer AND p.model = pce.model AND pce.price >= 500 AND pce.price <= 800}

2- {p.model | ∃pce ∈ PC : p.model = pce.model AND pce.manufacturer ≠ ∃Lap ∈ Laptop : Lap.manufacturer}

3- {l | ∃Lap ∈ Laptop: ∃Pri ∈ Printer: Lap.model = Pri.model AND Pri.is_colored = TRUE}

4- {p.manufacturer | ∃pce1, pce2 ∈ PC : pce1.speed > pce2.speed AND p.manufacturer = pce1.manufacturer}

SQL Query:

1- [SELECT manufacturer, model --- FROM PC --- WHERE price BETWEEN 500 AND 800;]

2- [SELECT model --- FROM PC --- WHERE manufacturer NOT IN (SELECT manufacturer FROM Laptop);]

3- [SELECT Laptop.model, Laptop.speed, Laptop.price --- FROM Laptop --- JOIN Printer ON Laptop.manufacturer = Printer.manufacturer AND Printer.is_colored = TRUE;]

4- [SELECT manufacturer --- FROM PC --- WHERE speed = (SELECT MAX (speed) FROM PC);]

(B):

Cooking Recipes Database

RA Expression:

1- π (food_item, grams, total_calories) (σ (recipe = "Mac & Cheese”) (ingredients ⋈ foodItem))

2- π (recipe) (σ (food_item = "Onions”) (ingredients) ⋈ σ (food_item = "Cheese”) (ingredients))

3- π (recipe) (ingredients - (σ (type = "Wheat”) (foodItem) ⋈ ingredients))

4- π (food_item, price) (σ (type = "Dairy" AND shop = "Carrefour”) (foodItem ⋈ stock))

RC Expression:

1- {<IN.food_item, IN.grams, SUM(IN.grams * F.calories_per_gram)> | IN ∈ ingredients, F ∈ foodItem, IN.food_item = F.food_item AND IN.recipe = "Mac & Cheese"}

2- {REC.recipe | ∃IN1, IN2 ∈ ingredients: IN1.recipe = REC.recipe AND IN2.recipe = REC.recipe AND IN1.food_item = "Onions" AND IN2.food_item = "Cheese"}

3- {REC.recipe | ∀IN ∈ ingredients: IN.recipe = REC.recipe AND ¬∃F ∈ foodItem : F.food_item = IN.food_item AND F.type = "Wheat"}

4- {<S.food_item, S.price> | S ∈ stock, F ∈ foodItem, F.food_item = S.food_item AND F.type = "Dairy" AND S.shop = "Carrefour"}

SQL Query:

1- [SELECT food_item, grams, SUM (grams * calories_per_gram) AS total_calories
FROM ingredients
JOIN foodItem ON ingredients.food_item = foodItem.food_item
WHERE recipe = "Mac & Cheese"
GROUP BY food_item, grams;]

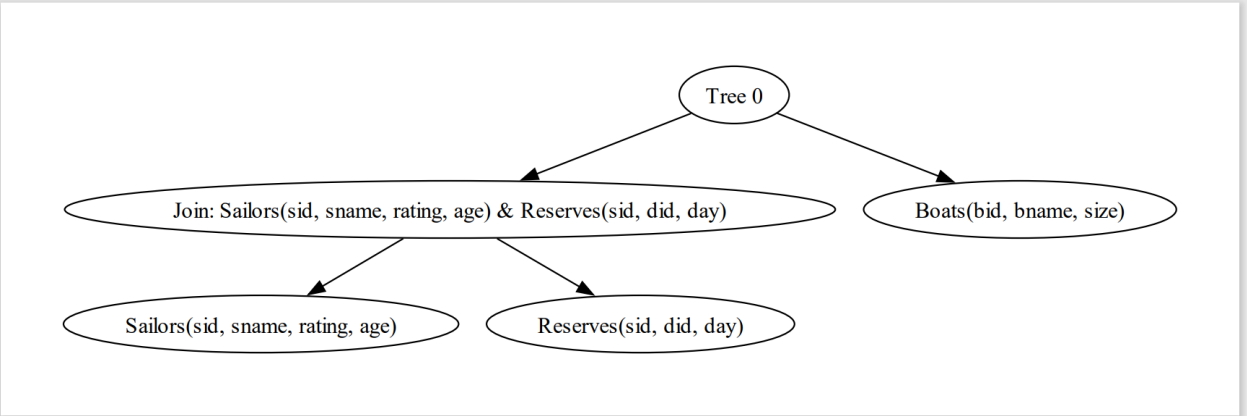
2- [SELECT recipe
FROM ingredients AS IN1
JOIN ingredients AS IN2 ON IN1. recipe = IN2.recipe
WHERE IN1.food_item = "Onions" AND IN2.food_item = "Cheese";]

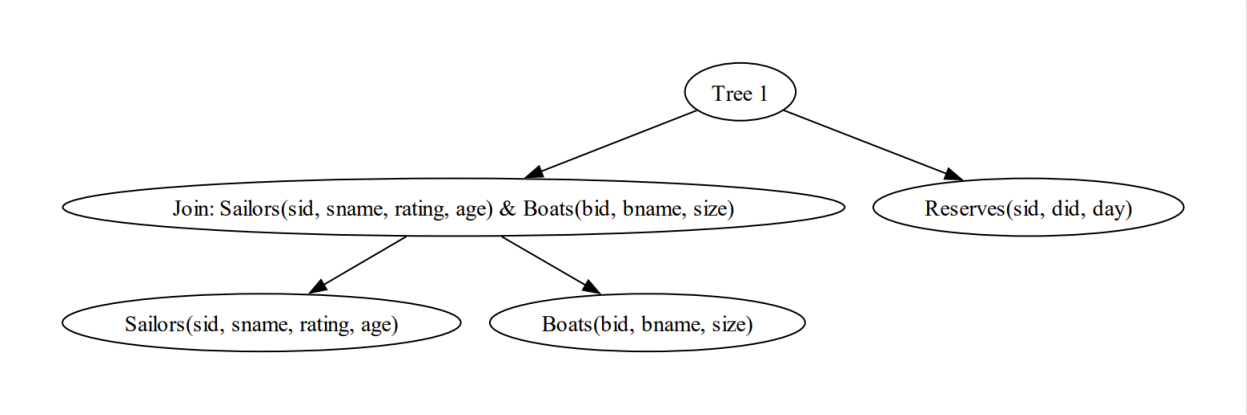
3- [SELECT recipe
FROM ingredients
WHERE recipe NOT IN (SELECT recipe FROM ingredients JOIN foodItem ON
ingredients.food_item = foodItem.food_item WHERE type = "Wheat");]

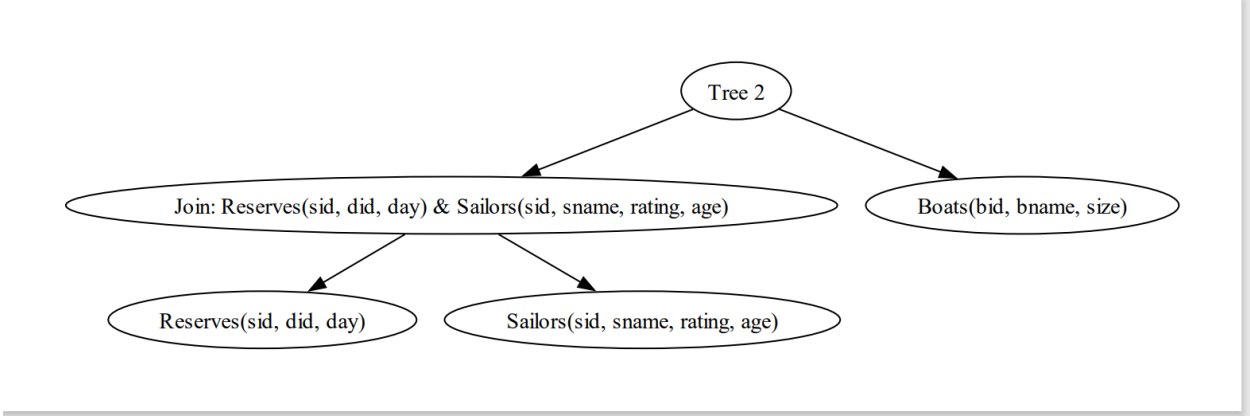
4- [SELECT food_item, prcie
FROM stock
JOIN foodItem ON stock.food_item = foodItem.food_item
WHERE type = "Dairy" AND shop = "Carrefour";]

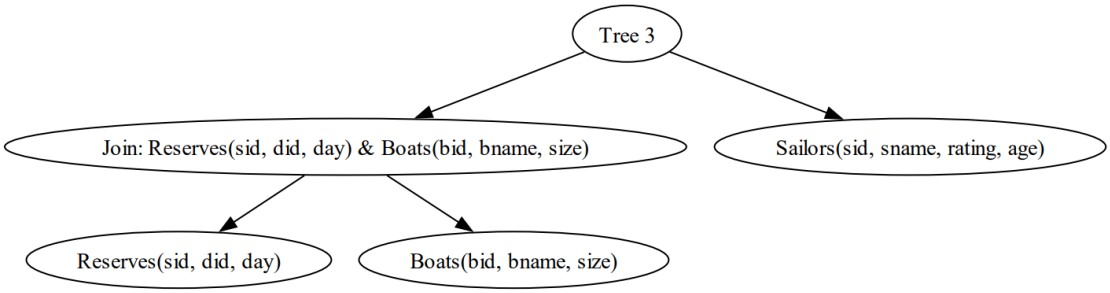
QUESTION (2):

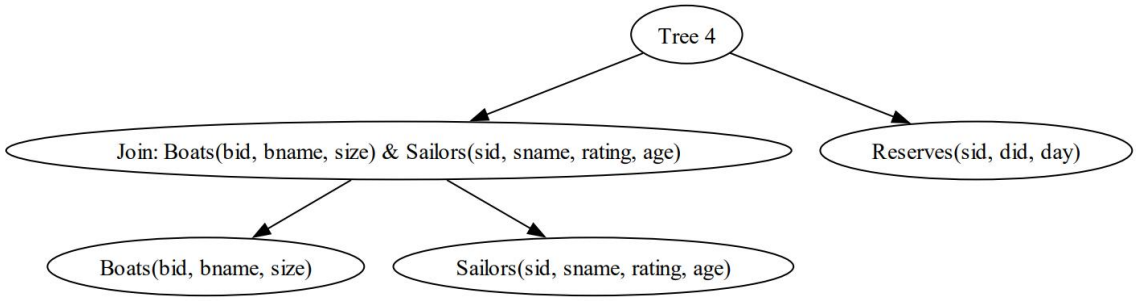
(A):

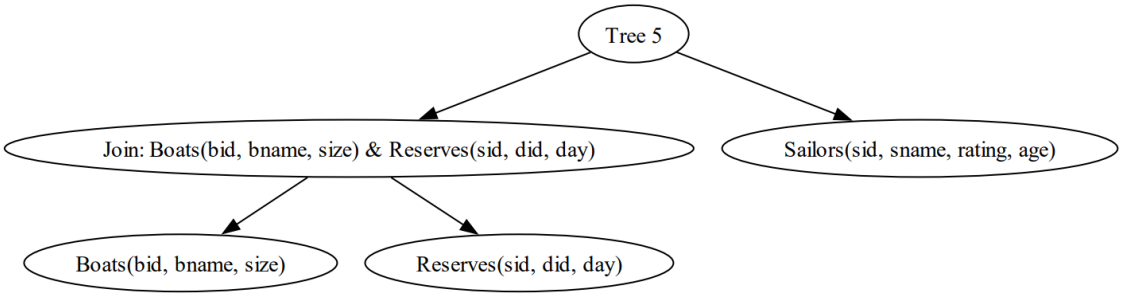

Tree 0
Join: Sailors(sid, sname, rating, age) & Reserves(sid, did, day)
Boats(bid, bname, size)
Sailors(sid, sname, rating, age)
Reserves(sid, did, day)


Tree 1
Join: Sailors(sid, sname, rating, age) & Boats(bid, bname, size)
Reserves(sid, did, day)
Sailors(sid, sname, rating, age)
Boats(bid, bname, size)


Tree 2
Join: Reserves(sid, did, day) & Sailors(sid, sname, rating, age)
Boats(bid, bname, size)
Reserves(sid, did, day)
Sailors(sid, sname, rating, age)


Tree 3
Join: Reserves(sid, did, day) & Boats(bid, bname, size)
Sailors(sid, sname, rating, age)
Reserves(sid, did, day)
Boats(bid, bname, size)


Tree 4
Join: Boats(bid, bname, size) & Sailors(sid, sname, rating, age)
Reserves(sid, did, day)
Boats(bid, bname, size)
Sailors(sid, sname, rating, age)


Tree 5
Join: Boats(bid, bname, size) & Reserves(sid, did, day)
Sailors(sid, sname, rating, age)
Boats(bid, bname, size)
Reserves(sid, did, day)

(B):

- Reserves contain 10,000 records with 40 records per page.
- Sailors contain 1000 records with 20 records per page.
- Boats contain 100 records with 10 records per page.

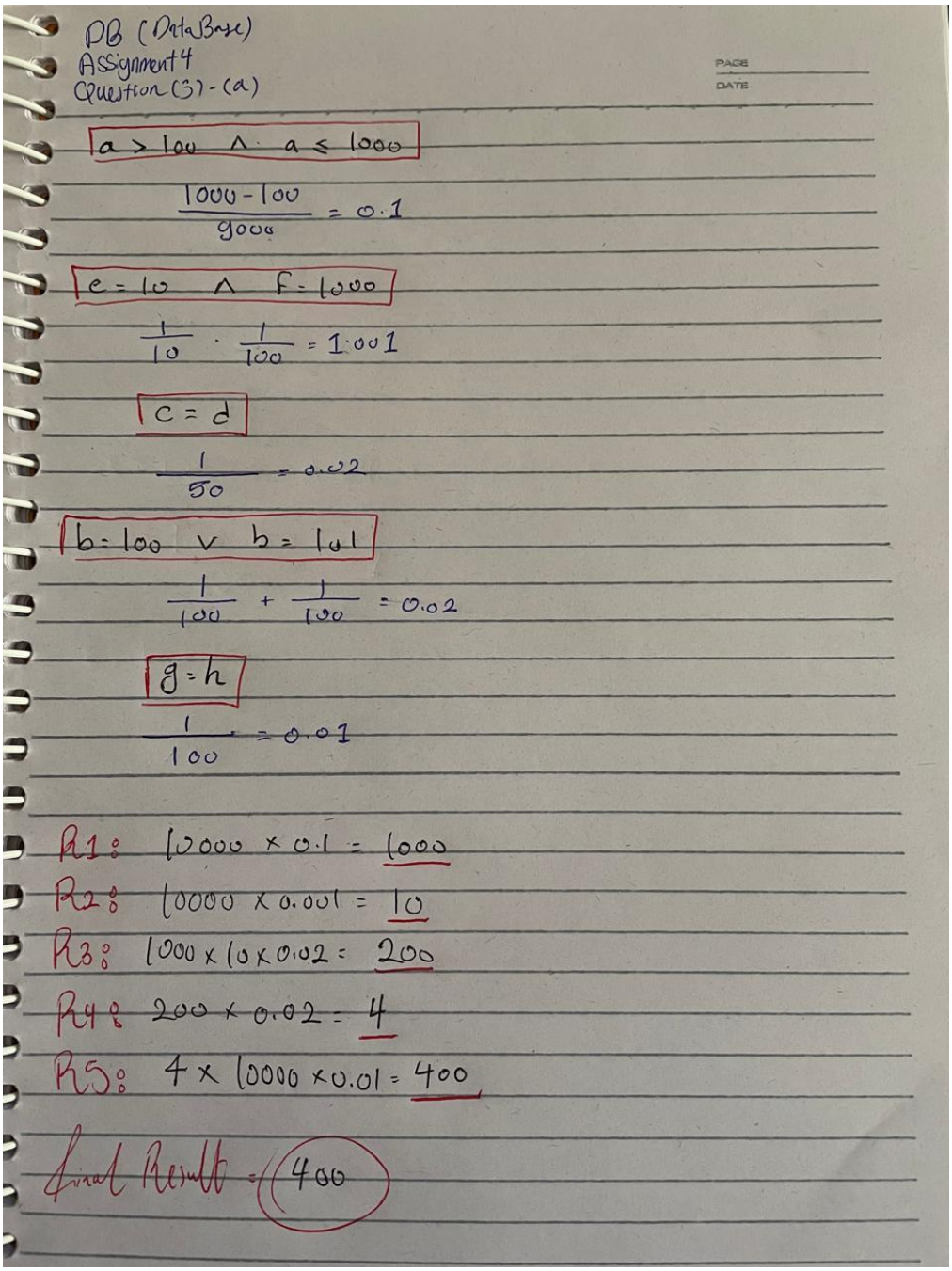
Page-based nested loop join and Block-based nested loop join are likely to perform the best for the first join, as they can utilize the buffer pool well enough, and in a mnimial way, by bringing pages or blocks of data into memory as needed. Given the buffer pool size and the relatively small sizes of the relations, both Reserves and Sailors are likely to fit entirely in memory, allowing efficient processing.

Sort-Merge Join may require significant disk overhead due to the need for external sorting.

Hash Join could also perform well for the smaller relation (Sailors), but may require additional disk for partitioning and probing the larger relation (Reserves) if it does not fit entirely in memory. However, it might still be efficient if the partitioning can be done effectively within the available buffer pool.

QUESTION (3):

(A):


DB (DataBase)
Assignment 4
Question (3) - (a)
PAGE
DATE
a > 100 ∧ a ≤ 1000
1000 - 100 / 9000 = 0.1
e = 10 ∧ f = 1000
1/10 · 1/100 = 1.001
c = d
1/50 = 0.02
b = 100 ∨ b = 101
1/100 + 1/100 = 0.02
g = h
1/100 = 0.01
R1: 10000 × 0.1 = 1000
R2: 10000 × 0.001 = 10
R3: 1000 × 10 × 0.02 = 200
R4: 200 × 0.02 = 4
R5: 4 × 10000 × 0.01 = 400
Final Result = 400

(B):

Cost = 4 * M * N2

Cost = 2.4 * N1

Cost = 2 * R disk reads + 2 * R disk writes

By calculating the costs of each on