**Laboratoire 1**

par

Alexandre Prud’Homme 7293804

Julien Desautels 7331746

Soumis à

Dr. Mok Beldjehem

dans le cadre du cours

Base de données II

CSI 3530

Université d’Ottawa

Le 11 novembre 2018

(A)

Les requêtes SQL pour chaque question :

SELECT sailors.sname, sailors.age

FROM SAILORS;

SELECT DISTINCT sailors.sname, sailors.age

FROM SAILORS;

SELECT sailors.sid, sailors.sname, sailors.rating, sailors.age

FROM SAILORS

WHERE sailors.rating > 7;

SELECT sailors.sname

FROM SAILORS

INNER JOIN RESERVES ON sailors.sid = reserves.sid

WHERE reserves.bid = 103;

SELECT sailors.sname

FROM SAILORS

INNER JOIN RESERVES ON sailors.sid = reserves.sid

INNER JOIN BOATS ON reserves.bid = boats.bid

WHERE boats.color = 'red'

GROUP BY sailors.sid;

SELECT DISTINCT boats.color

FROM BOATS

INNER JOIN RESERVES ON boats.bid = reserves.bid

INNER JOIN SAILORS ON reserves.sid = sailors.sid

WHERE sailors.sname = 'Lubber';

SELECT sailors.sname

FROM sailors

INNER JOIN reserves ON sailors.sid = reserves.sid

GROUP BY sailors.sid;

SELECT sailors.sname, sailors.rating + 1

FROM sailors

INNER JOIN reserves r1 ON sailors.sid = r1.sid

INNER JOIN reserves r2 ON sailors.sid = r2.sid

WHERE r1.day = r2.day AND r1.bid <> r2.bid;

SELECT sailors.age

FROM sailors

WHERE (position('B' in sailors.sname) = 1) and

(position ('b' in sailors.sname) = 3) and

(char\_length(sailors.sname) = 3);

SELECT sailors.sname

FROM sailors

INNER JOIN reserves ON sailors.sid = reserves.sid

INNER JOIN boats ON reserves.bid = boats.bid

WHERE (boats.color = 'red') or (boats.color = 'green')

GROUP BY sailors.sid;

SELECT s1.sname

FROM sailors s1, reserves r1, boats b1

WHERE s1.sid = r1.sid AND r1.bid = b1.bid AND b1.color = 'red'

INTERSECT

SELECT s2.sname

FROM sailors s2, reserves r2, boats b2

WHERE s2.sid = r2.sid AND r2.bid = b2.bid AND b2.color = 'green';

SELECT s1.sid

FROM sailors s1, reserves r1, boats b1

WHERE s1.sid = r1.sid AND r1.bid = b1.bid AND b1.color = 'red'

EXCEPT

SELECT s2.sid

FROM sailors s2, reserves r2, boats b2

WHERE s2.sid = r2.sid AND r2.bid = b2.bid AND b2.color = 'green';

SELECT sailors.sid

FROM sailors

WHERE sailors.rating = 10

UNION

SELECT reserves.sid

FROM reserves

WHERE reserves.bid = 104;

SELECT AVG(sailors.age)

FROM sailors;

SELECT AVG(sailors.age)

FROM sailors

WHERE sailors.rating = 10;

SELECT sailors.sname, sailors.age

FROM sailors

WHERE sailors.age = (SELECT MAX(sailors.age)

FROM sailors);

SELECT COUNT(sailors.sid)

FROM sailors;

SELECT COUNT(DISTINCT sailors.sname)

FROM sailors;

SELECT MIN(sailors.age), sailors.rating

FROM sailors

GROUP BY sailors.rating;

SELECT MIN(sailors.age), sailors.rating

FROM sailors

WHERE sailors.age >= 18

GROUP BY sailors.rating

HAVING COUNT(sailors) >= 2;

(B)

Les plans d’évaluation pour chaque requête SQL :









































(C)

Les chemins d’accès et les méthodes d’implémentations des opérateurs impliqués dans chaque plan d’évaluation pour chaque requête SQL :

1. Un scan séquentiel est utilisé sur la table « sailors » et un retour de « sname » et « age ». « EXPLAIN ANALYZE » démontre ceci avec un temps d’exécution de 0.775 ms.

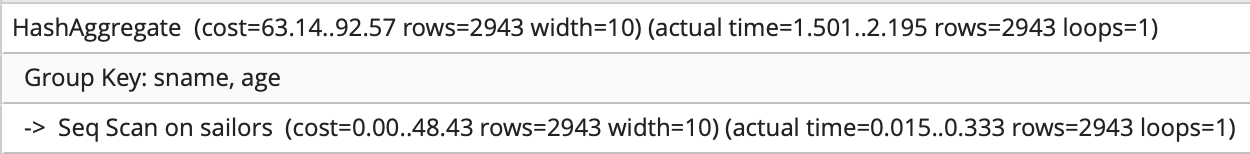


Planning time : 0.159 ms

Execution time : 0.775 ms

1. Un agrégat de hachage est utilisé afin d’éliminer les tuples identiques. L’agrégat de hachage a besoins d’un opérateur d’agrégat et une clé en groupe. Dans ce cas, la clé en groupe est « sname » et « age ». Un scan séquentiel est utilisé et le retour de l’information est fait.

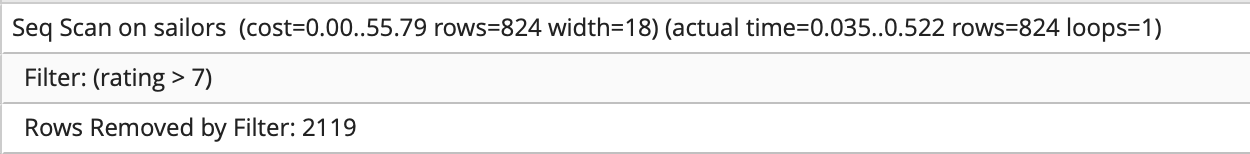
« EXPLAIN ANALYZE » démontre ceci avec un temps d’exécution de 2.546 ms.



Planning time : 0.124 ms

Execution time : 2.546 ms

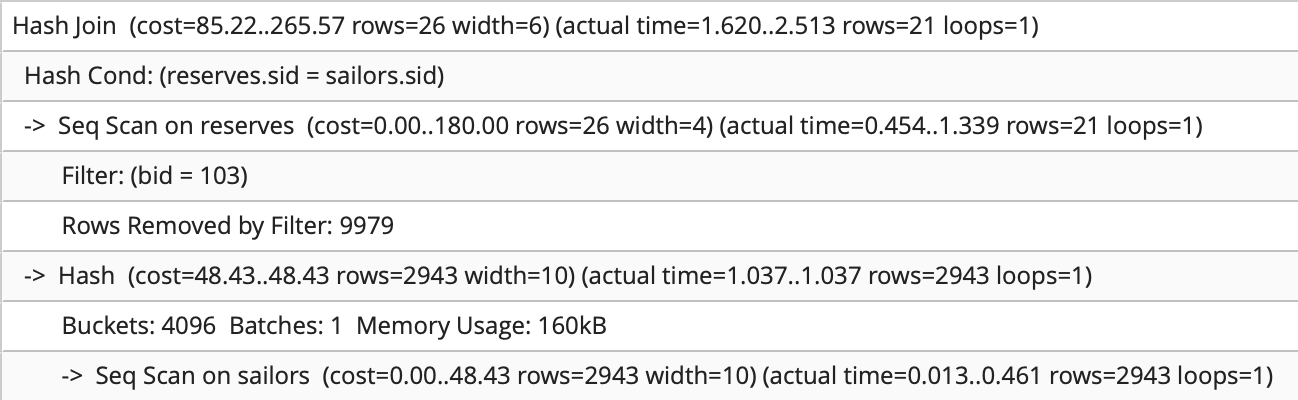
1. Un scan séquentiel avec un filtre « rating > 7 » est utilisé pour retourner l’information. « EXPLAIN ANALYZE » démontre ceci avec un temps d’exécution de 0.613 ms.



Planning time : 0.360 ms

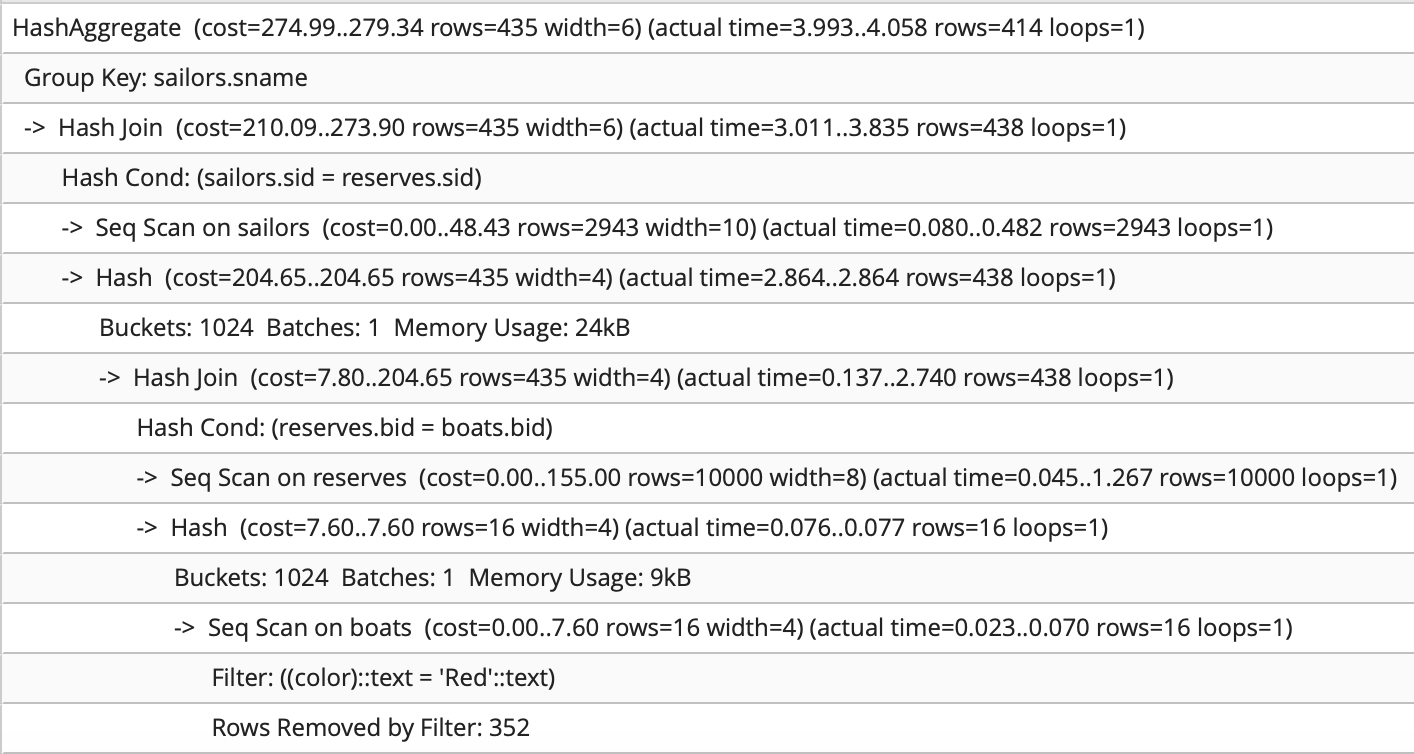
Execution time : 0.613 ms

1. Avec la condition de hachage « sailors.sid = reserves.sid » créer une jointure de hachage. Par la suite, faire un scan séquentiel sur la table « reserves » avec le filtre « reserves.bid = 103 ». Avec le hash faire un scan séquentiel de « sailors » pour retourner l’information. « EXPLAIN ANALYZE » démontre ceci avec un temps d’exécution de 2.592 ms.



Planning time : 0.400 ms

Execution time : 2.592 ms



Planning time : 0.677 ms

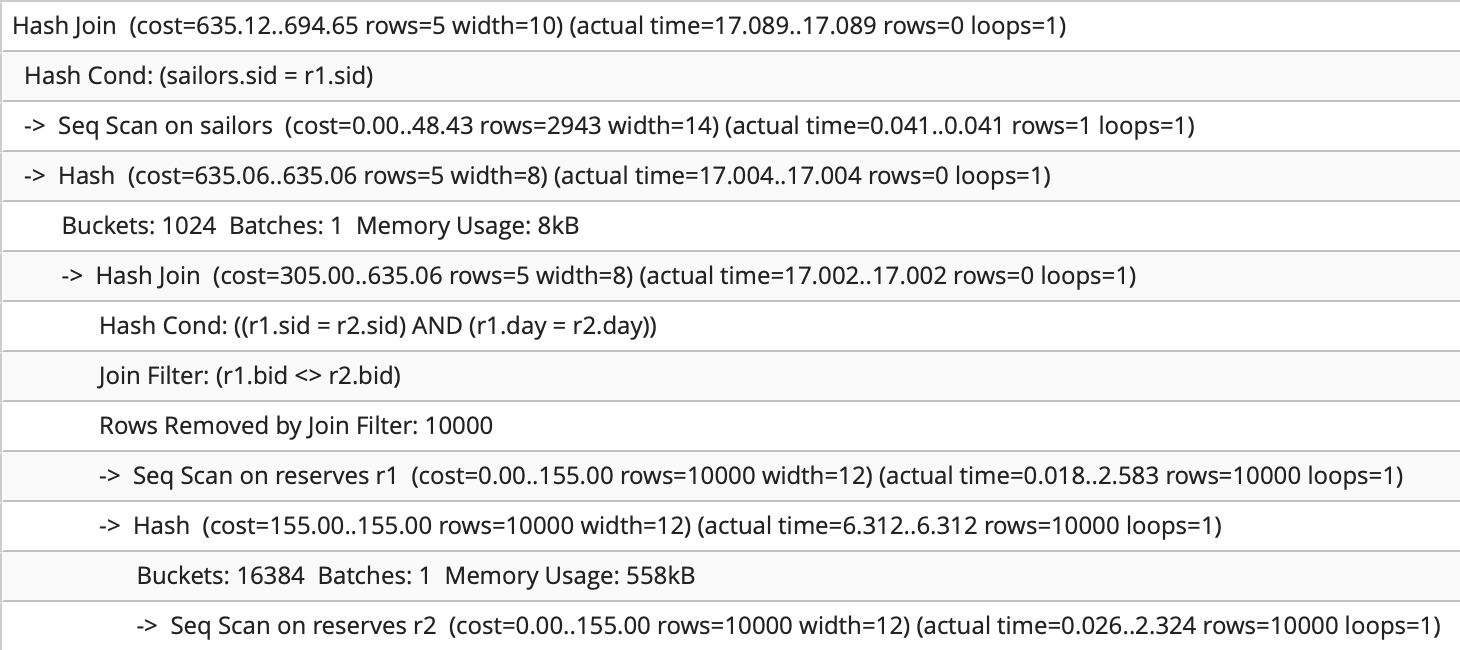
Execution time : 4.372 ms



Planning time : 0.408 ms

Execution time : 0.924 ms

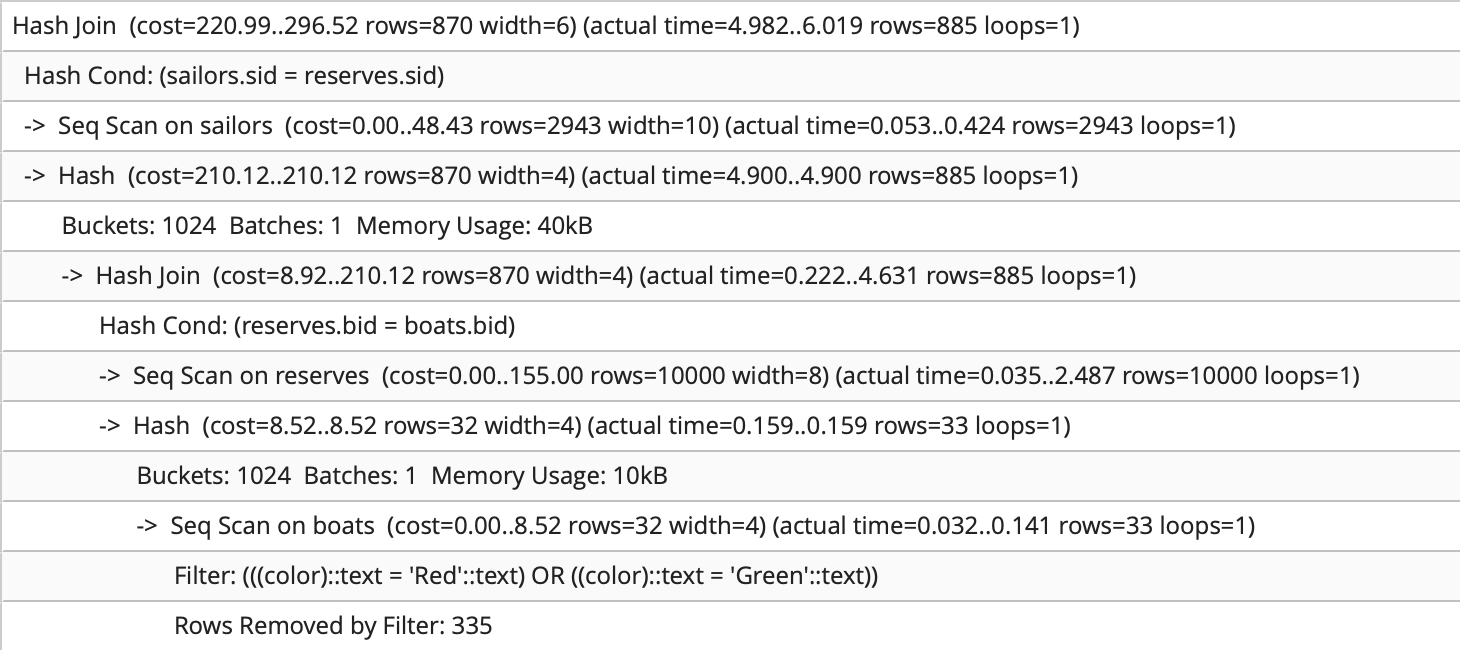
1. Fix



Planning time : 1.209 ms

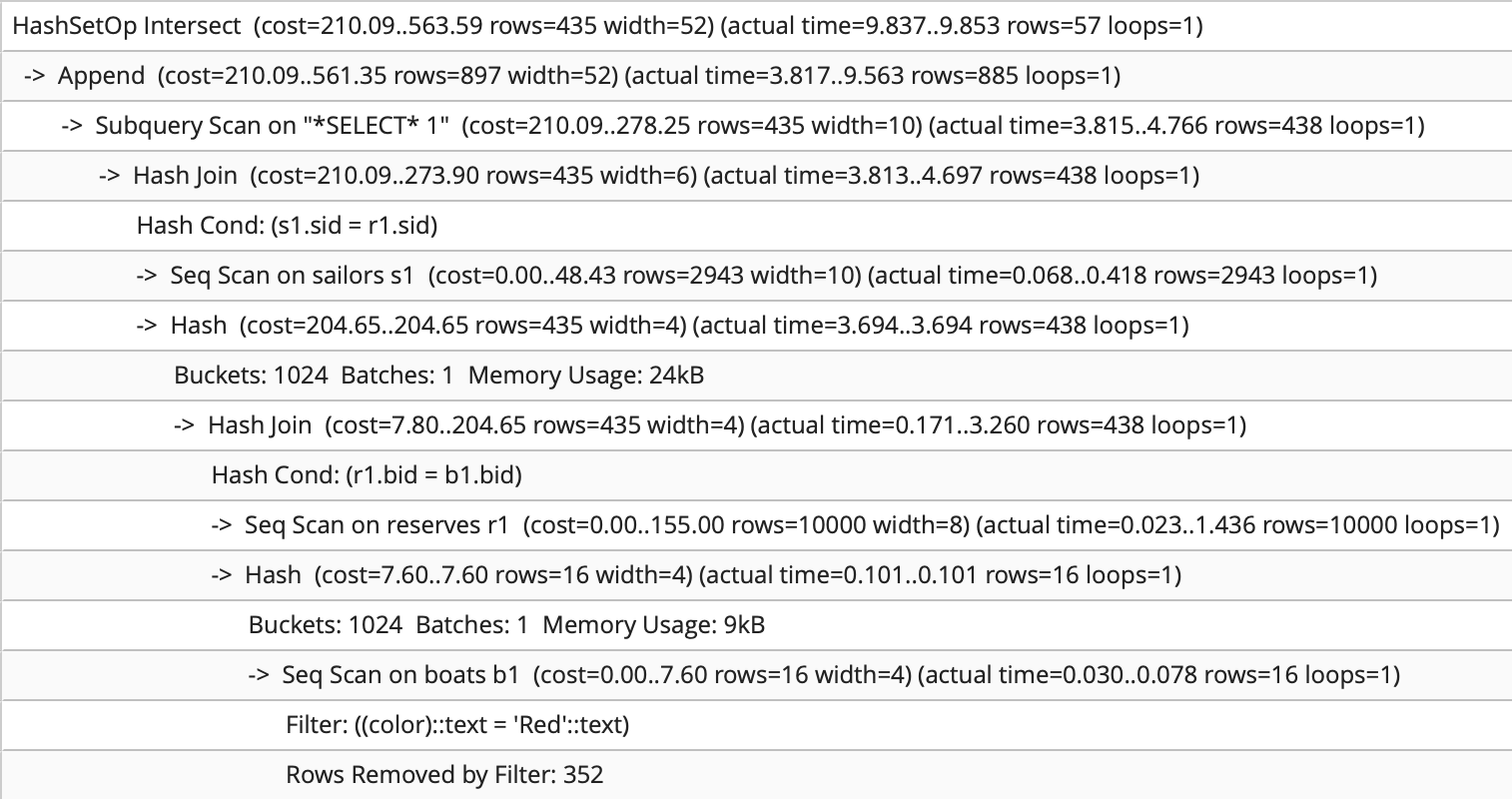
Execution time : 17.329 ms

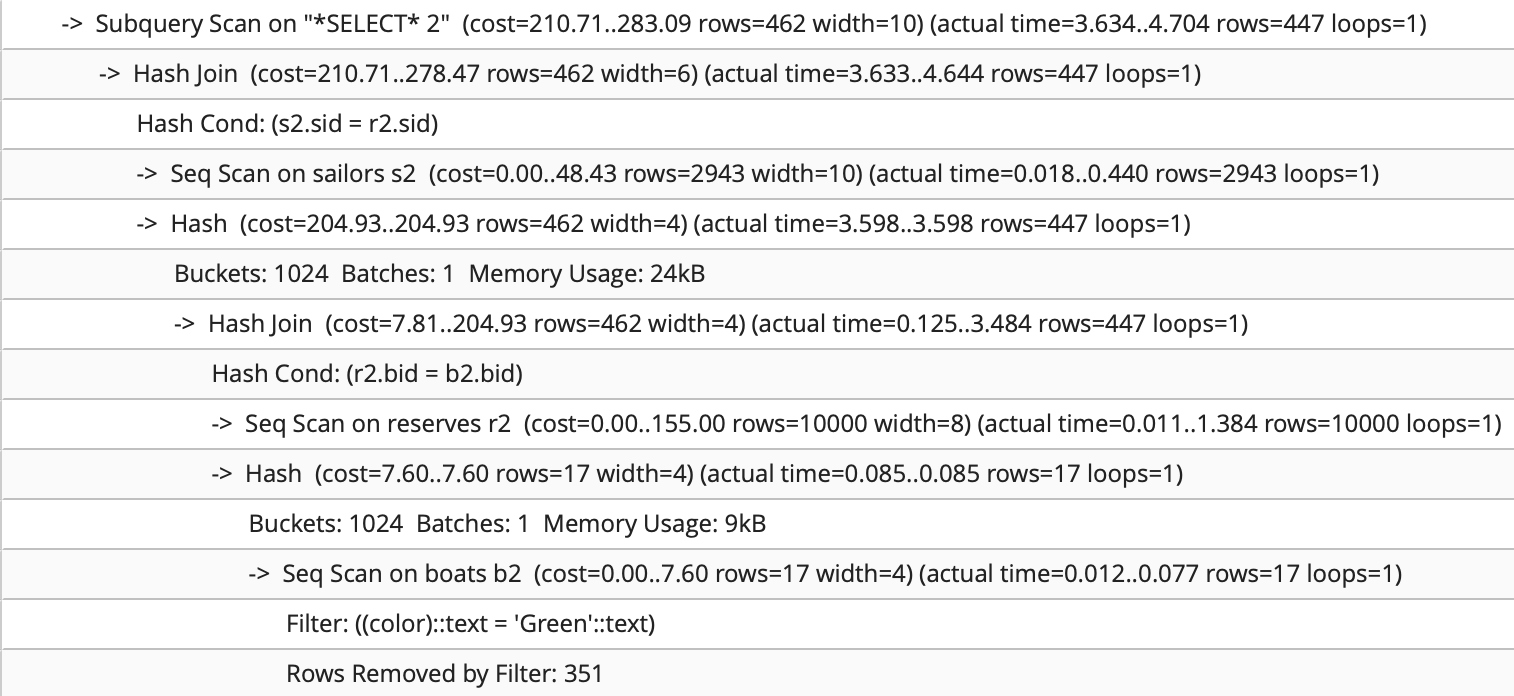
1. FIX



Planning time : 0.224 ms

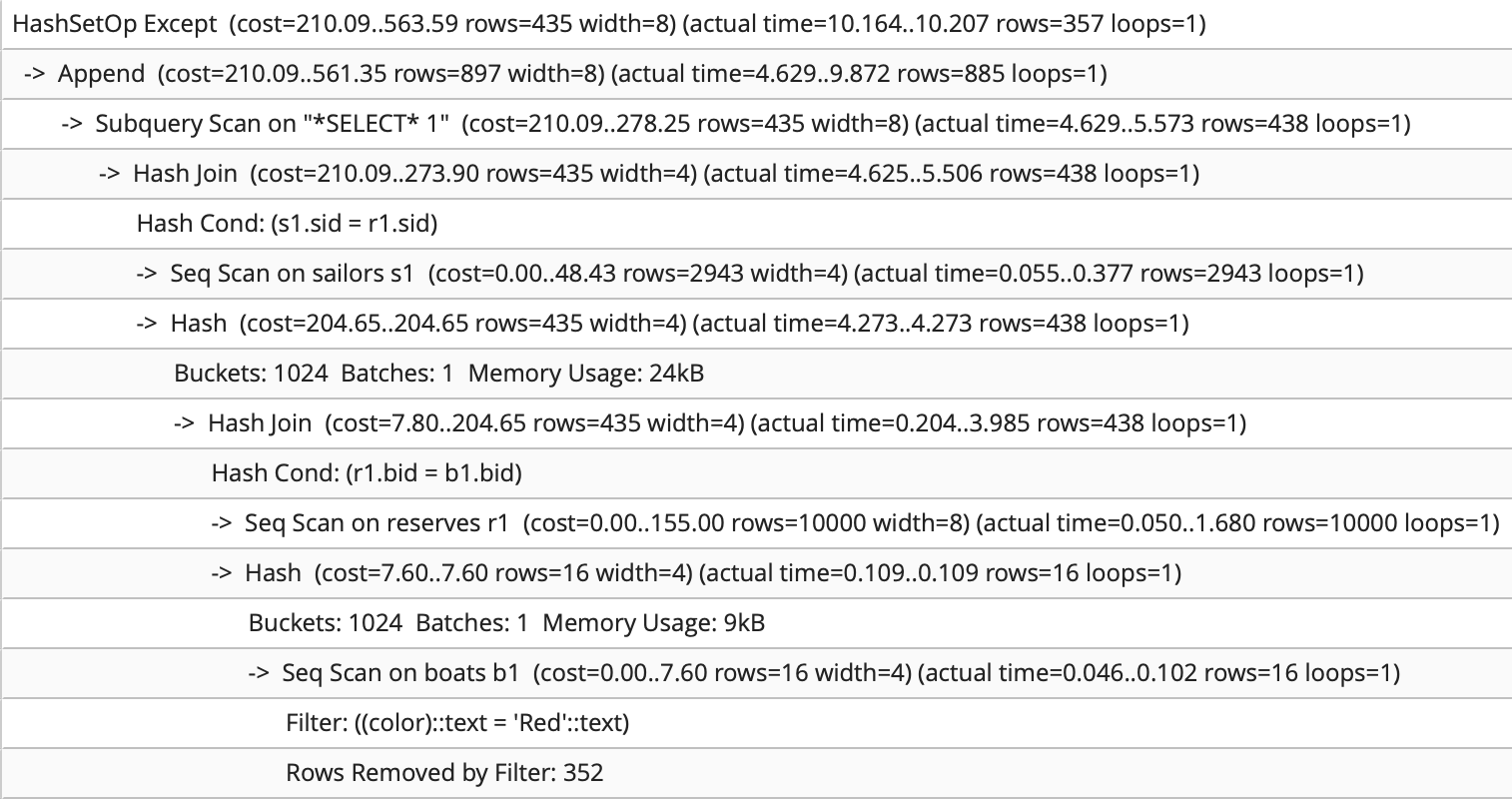
Execution time : 6.118 ms

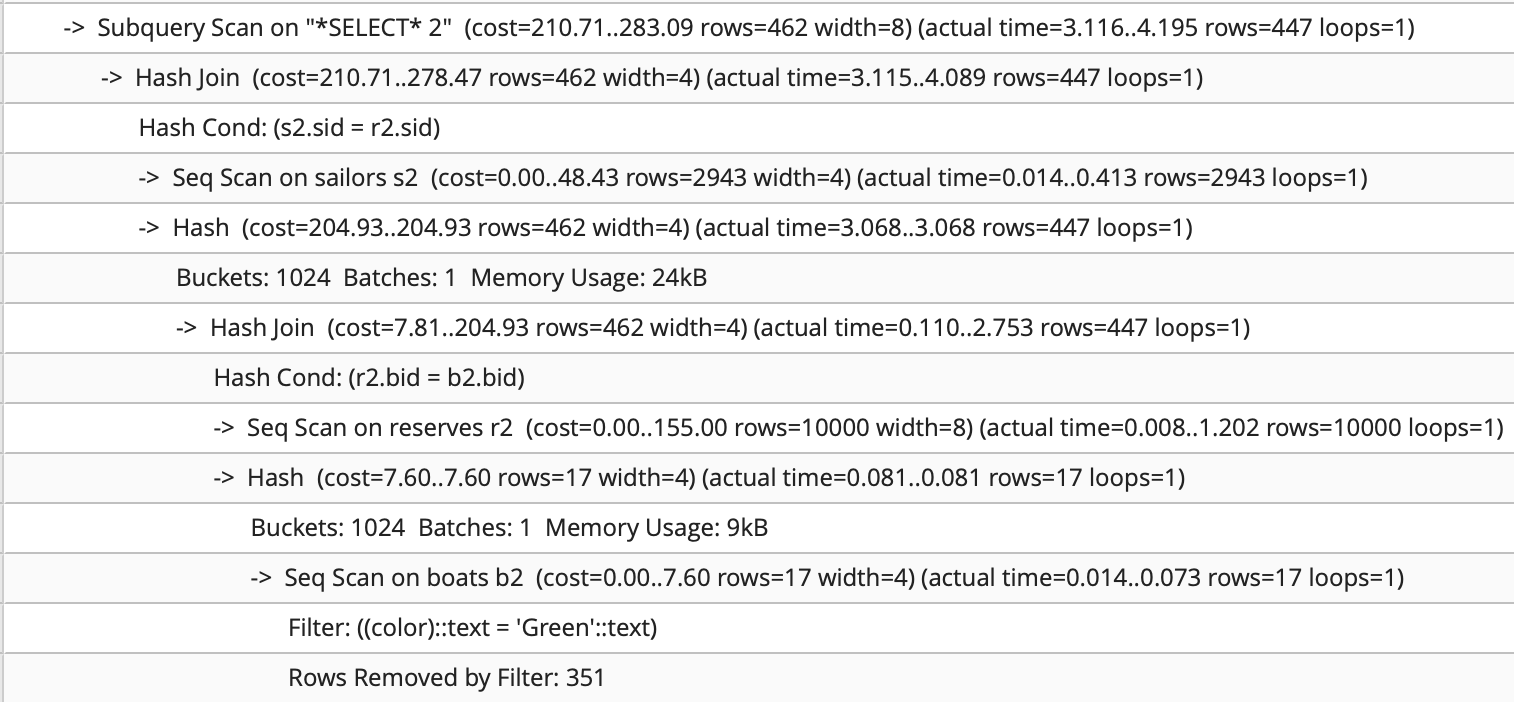




Planning time : 0.810 ms

Execution time : 10.105 ms

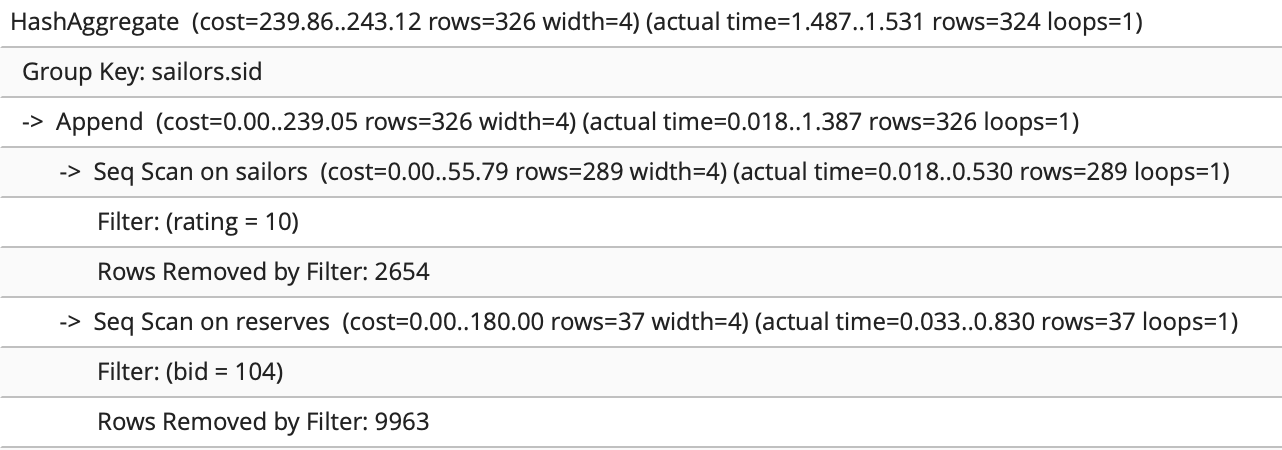




Planning time : 0.532 ms

Execution time : 10.757 ms

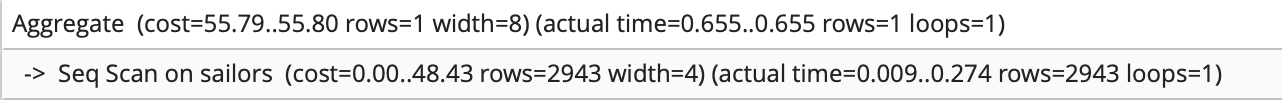
1. Un agrégat de hachage avec une clé en groupe « sailors.sid » est utilisé. Par la suite, il y a l’union entre le scan séquentiel de « sailors » avec le filtre « sailors.rating = 10 » et le scan séquentiel de « reserves » avec le filtre « reserves.bid = 104 ». « EXPLAIN ANALYZE » démontre ceci avec un temps d’exécution de 1.575 ms.



Planning time : 0.090 ms

Execution time : 1.575 ms

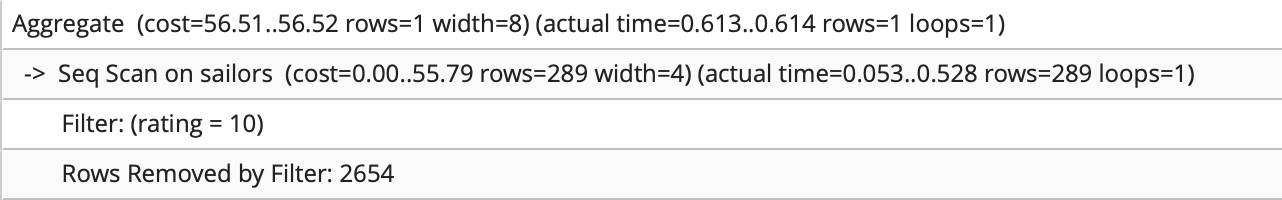
1. L’agrégat de la moyenne est utilisé. Alors, un scan séquentiel de « sailors » est utilisé pour retourner la moyenne de l’âge. « EXPLAIN ANALYZE » démontre ceci avec un temps d’exécution de 0.696 ms.



Planning time : 0.115 ms

Execution time : 0.696 ms

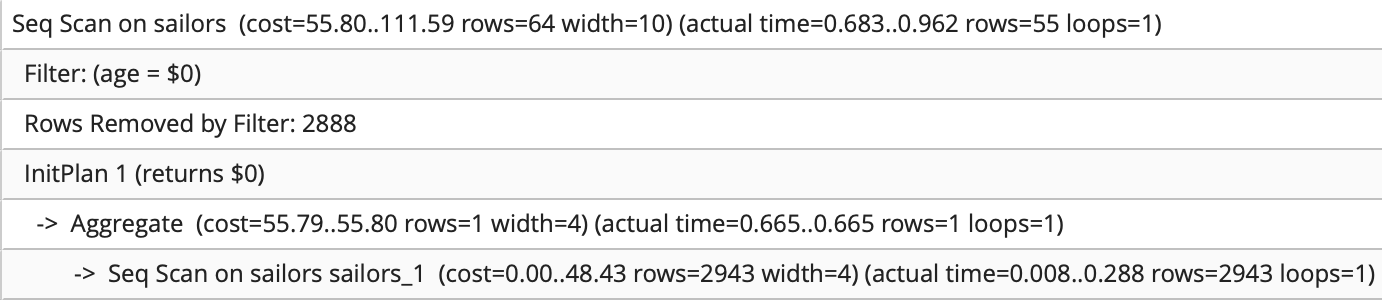
1. L’agrégat de la moyenne est utilisé. Alors, un scan séquentiel de « sailors » est utilisé avec un filtre de « sailors.rating = 10 » pour retourner la moyenne de l’age. « EXPLAIN ANALYZE » démontre ceci avec un temps d’exécution de 0.716 ms.



Planning time : 0.768 ms

Execution time : 0.716 ms

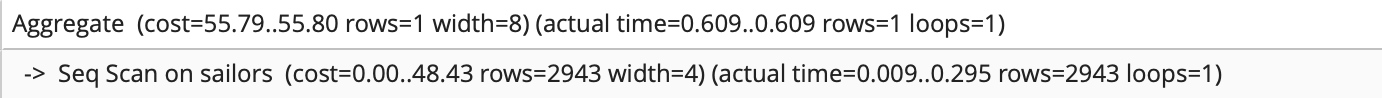
1. La requête est de forme imbriquée. Alors, il y a un scan séquentiel de la table « sailors » avec filtre. Le filtre est la requête imbriquée. Alors, cette requête imbriquée utilise l’agrégat du maximum de l’âge selon le scan séquentiel de la table « sailors ». « EXPLAIN ANALYZE » démontre ceci avec un temps d’exécution de 1.002 ms.



Planning time : 0.319 ms

Execution time : 1.002 ms

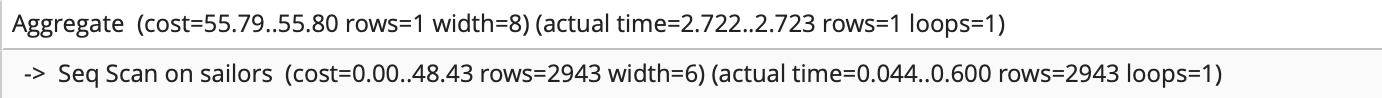
1. L’agrégat de la somme est utilisé de « sailors.sid ». Alors, un scan séquentiel de la table « sailors » est utilisé pour calculer la somme. « EXPLAIN ANALYZE » démontre ceci avec un temps d’exécution de 0.640 ms.



Planning time : 0.063 ms

Execution time : 0.640 ms

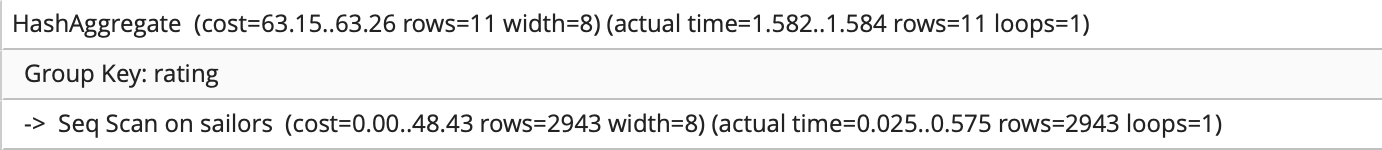
1. L’agrégat de la somme est utilisé de « sailors.sname ». Selon « distinct » les tuples en doubles sont supprimés lors du scan séquentiel de la table « sailors ». « EXPLAIN ANALYZE » démontre ceci avec un temps d’exécution de 2.800 ms.



Planning time : 0.097 ms

Execution time : 2.800 ms

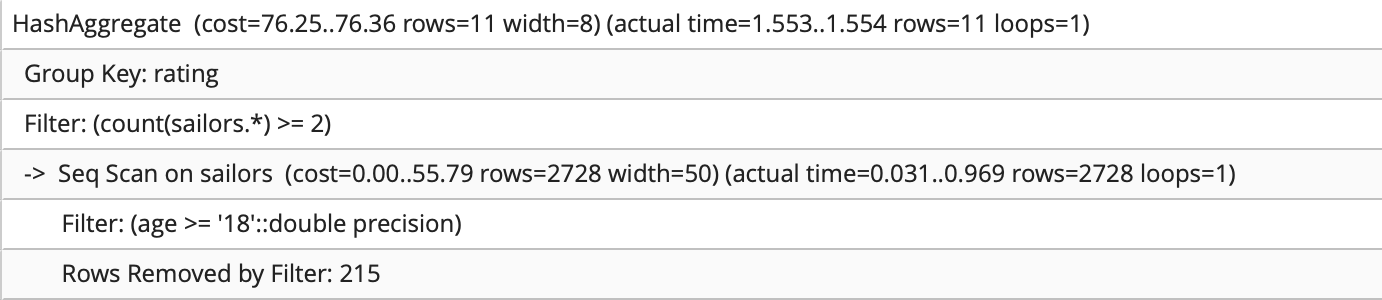
1. L’agrégat de hachage est utilisé avec la clé en groupe « sailors.rating ». L’agrégat du minimum est utilisé pour « sailors.age ». Alors, il y a un scan séquentiel de la table « sailors » afin de retourner l’information. « EXPLAIN ANALYZE » démontre ceci avec un temps d’exécution de 1.645 ms.



Planning time : 0.150 ms

Execution time : 1.645 ms

1. L’agrégat de hachage est utilisé avec la clé en groupe « sailors.rating » et le filtre de la somme des « sailors » qui est supérieur ou égal à deux. Par la suite, un scan séquentiel avec le filtre « age >= 18 » est appliqué à la table « sailors ». « EXPLAIN ANALYZE » démontre ceci avec un temps d’exécution de 1.586 ms.



Planning time : 0.152 ms

Execution time : 1.586 ms