Advanced Databases: improvements

C4

Samuel Lúcio Vicente, 251720

Daniel Silva, 251702

Jorge Marrero Camiruaga, 251438

1 Improvement (Query 1)

Improvement:

- B-Tree index on Account based on the amount.

Type:

- B-Tree index.

Purpose:

- Have account ordered by amount that way the max value is easily calculated and the number of rows visited until the average amount is found is only the necessary.

```
UPDATE Account
SET amount = amount*2
WHERE amount <= ALL(
       SELECT Avg(amount)
       FROM Account
```

```
SELECT Name
    FROM Costumer
    INNER JOIN Person
        ON GovID = PersonGovID
    INNER JOIN Account
        ON CostumerID = CostumerCostumerID
    WHERE amount >= ALL(
        SELECT MAX(amount)
```

2 Improvement (Query 1)

Improvement:

- Partition Account by range of amount.

Type:

- Partition by range.

Purpose:

- Have account partitioned by amount that way the max value is more easily calculated as it only accesses the higher partition and has less rows to search. The average still has to access every row but now the number of rows visited until the average amount is found is less because the partitions whose range is bigger than the average are not fetched .

```
UPDATE Account
SET amount = amount*2
WHERE amount <= ALL(
        SELECT Avg(amount)
        FROM Account
```

```
SELECT Name
    FROM Costumer
    INNER JOIN Person
        ON GovID = PersonGovID
    INNER JOIN Account
        ON CostumerID = CostumerCostumerID
    WHERE amount >= ALL(
        SELECT MAX(amount)
```

3 Improvement (Query 2)

Improvement:

- List partition in Person based on the nationalities.
- Range partition in SavingsAccount based on duration years, one partition for <7 and one for >7.

Type:

- Partition by List
- Partition by Range

Purpose:

- The where clause on the select query now only fetches the partition whose nationality is Poland, less reads from memory and less comparisons.
- The where clause on the select query also gets improved as now only the SavingsAccounts whose durationyears>7 gets fetched, less reads from memory and less comparisons.
- The update query also gets improved as now there are less comparisons being made.

```
select count(*)
from savingsaccount inner join Account
    on AccountAccountID = AccountID
inner join Costumer
    on CostumerCostumerID = CostumerID
inner join Person
    on PersonGovID = GovID
where durationyears>7 and Nationality = 'Poland';
```

```
update SavingsAccount
set interestRate =
CASE
    WHEN durationYears>7 THEN interestrate + 0.03
    ELSE interestrate + 0.01
END;
```

4 Improvement (Query 3)

Improvement:

- Allocate SavingAccount and CurrentAccount in same disk and Account in another disk.

Type:

- Partition data using external memory.

Purpose:

- Enable parallel reading of both tables for a faster join time.

```
from Account inner join SavingsAccount
from Account inner join CurrentAccount
```

5 Improvement (Query 4)

Improvement:

- Hash partition RoleHistory by EmployeeEmployeeID and separate them in different discs.

Type:

- Hash partition using external memory

Purpose:

- Enable parallel reading of multiple rows.

```
(SELECT BranchBranchID FROM RoleHistory WHERE EmployeeEmployeeID = empID AND EndDate is NULL),
```

```
WHERE HistoryID = (SELECT min(HistoryID) FROM RoleHistory WHERE EmployeeEmployeeID = empID AND
    ↪ EndDate is NULL);
```

6 Improvement (Query 4)

Improvement:

- BitMap index in RoleHistory based in EndDate of RoleHistory if null or not.

Type:

- BitMap Index.

Purpose:

- Faster times fetching the rows that have the EndDate null and less comparisons.

```
(SELECT BranchBranchID FROM RoleHistory WHERE EmployeeEmployeeID = empID AND EndDate is NULL),

  WHERE HistoryID = (SELECT min(HistoryID) FROM RoleHistory WHERE EmployeeEmployeeID = empID AND
      ↪ EndDate is NULL);
```

7 Improvement (Query 4)

Improvement:

- List partition in RoleHistory based on the EndDate being null or not.
- B-Tree index based on the EmployeeEmployeeID.

Type:

- Partition by List.
- B-Tree index.

Purpose:

- Less reads and less comparisons for the null EndDate as it is now all stored in a partition and faster fetch times for the EmployeeEmployeeID as it is stored in a B-Tree inside the Partition.

```
(SELECT BranchBranchID FROM RoleHistory WHERE EmployeeEmployeeID = empID AND EndDate is NULL),
```

```
WHERE HistoryID = (SELECT min(HistoryID) FROM RoleHistory WHERE EmployeeEmployeeID = empID AND
    ↪ EndDate is NULL);
```