Bases de Dados

Lab 11: Indexes & Optimisation

Default Indexes

1. Let's populate the **account** table by running the `\i index_data.sql` in the psql shell.

2. To obtain information about the indexes of the **account** table, run the command `\d account` . The system seems to have created an index for the table's primary key by default. What kind?

3. Enable automatic execution time reporting in the psql shell with the command `\timing.`

4. Now when you run the following query:

```
SELECT * FROM account WHERE account_number='A-012345';
```

   You should see first the results followed by the time it took the system produce them

5. Now lets delete the primary key (along with the index that was created by default):

```
ALTER TABLE account DROP CONSTRAINT account_pkey;
```

6. Repeat **step 4** and take note of the time. How do you explain this result?

7. So, let's add back the primary key:

```
ALTER TABLE account ADD PRIMARY KEY(account_number);
```

   Notice it took some time to execute this command. Why?

8. Repeat **step 4** and note the general trend in terms of execution time.

Index Creation

9. Let's run `\i index_data.sql` again to populate the account table again and clear any changes.

10. Run the query and note its execution time

```
SELECT account_number FROM account WHERE balance=1000;
```

11. Run the query and note its execution time

```
SELECT MAX(balance) FROM ACCOUNT;
```

12. Now let's create an index for the column "balance" used in both queries with the command:

```
CREATE INDEX balance_idx ON account(balance);
```

Is this a primary index or a secondary index?

13. To obtain information about the indexes of the account table, run the command `\d account`. The system created an index for the balance column. What kind?

14. Now repeat the queries from **step 10 and 11** and note their new execution time. For both queries, how do you explain the time difference?

15. Now delete the index created previously in **step 12**

```
DROP INDEX balance_idx;
```

16. Let's explicitly create a HASH index for the column balance, specifying the type of index like so:

```
CREATE INDEX balance_idx ON account USING HASH(balance);
```

17. Now repeat the queries from **step 10 and 11** and note their new execution time. For both queries, how do you explain the time difference?

18. You can now delete the index created in **step 16**:

```
DROP INDEX balance_idx;
```

Execution Plans

19. Let's run **`\i index_data.sql`** again to populate the account table again and clear any changes.

20. Get the execution plan for the query of **step 11** with the command:

```
EXPLAIN SELECT MAX(balance) FROM ACCOUNT;
```

What access method is used? Justify.

21. TIP: You can estimate the execution-time more accurately running ANALYZE (excludes network)

```
EXPLAIN ANALYZE SELECT MAX(balance) FROM ACCOUNT;
```

From the manual: *The ANALYZE option causes the statement to be actually executed, not only planned. The total elapsed time expended within each plan node (in milliseconds) and total number of rows it actually returned are added to the display. This is useful for seeing whether the planner's estimates are close to reality.*

22. Now create a B+TREE index on the balance attribute and check the access plan again:

```
CREATE INDEX balance_idx ON account(balance);

EXPLAIN SELECT MAX(balance) FROM ACCOUNT;
```

What difference do you see in the access method?

23. Create a HASH index for the balance column, compare the access plan with **step 20**

```
DROP INDEX balance_idx;

CREATE INDEX balance_idx ON account USING HASH(balance);

EXPLAIN SELECT MAX(balance) FROM ACCOUNT;
```

How do you explain that the hash index is never used?

Query Optimisation

24. Consider the following table:

```
CREATE TABLE employee (
  eid INTEGER PRIMARY KEY,
  ename VARCHAR(40) NOT NULL,
  address VARCHAR(255) NOT NULL,
  salary NUMERIC(12,4) NOT NULL,
  bdate DATE NOT NULL);
```

Think about the indexes you could/should create to improve the efficiency of frequently executed queries. For this exercise, let's suppose that the following queries are quite common:

a) What is the identifier, name, and address of employees aged within a certain range?

b) What is the identifier and address of employees with a given name?

c) What is the maximum salary for employees?

d) What is the average salary of employees by age?

TIP: Write down the SQL queries to think about which indices would be more advantageous.