Assignment 3

Index Tuning

Database Tuning

New Group 8
Frauenschuh Florian, 12109584
Lindner Peter, 12101607
Weilert Alexander, 12119653

April 30, 2024

Database system and version: `Postgres 14.11` with driver `postgresql 42.7.3`

1 Index Data Structures

Which index data structures (e.g., B^+ tree index) are supported?

- B-Tree [1]
  - Default when using `CREATE INDEX`
  - `CREATE INDEX index_name ON table_name (column_name);`
- Hash [1]
  - Hash index stores 32-bit hash derived from indexed column
  - Only supports equality comparisons
  - `CREATE INDEX index_name ON table_name USING HASH (column_name);`
- GiST [1] [2]
  - "Generalized Search Tree"
  - Infrastructure of many index strategies
  - Lossy index (may produce false matches)
  - `CREATE INDEX index_name ON table_name USING gist(column_name);`
- SP-GiST [1]
  - "Space-partitioned GiST"
  - Non-balanced data structures
  - `CREATE INDEX index_name ON table_name USING spgist(column_name);`
- GIN [1] [2]

  - "Generalized Inverted Index"
  - Supports many different indexing strategies
  - Appropriate for values containing multiple component, e.g.: arrays

    ```
    CREATE INDEX index_name ON table_name USING gin(column_name);
    ```

- BRIN [1]
  - "Block Range INdex"
  - Indexes ranges of values
  - `CREATE INDEX index_name ON table_name USING brin(column_name);`

2 Clustering Indexes

Discuss how the system supports clustering indexes, in particular:

a) How do you create a clustering index on `ssnum`? Show the query.[1]

First, we assume the table `Employee` to be the same as in the previous assignment:

```
CREATE TABLE IF NOT EXISTS Employee (
    ssnum INTEGER PRIMARY KEY,
    name VARCHAR(64) UNIQUE NOT NULL,
    manager VARCHAR(64),
    dept VARCHAR(64),
    salary INTEGER,
    numfriends INTEGER);
```

We note that `ssnum` is the primary key of the table, hence Postgres automatically creates an B-Tree based, *unique index* for it [3].

Now to be able to create a clustering index, we need an index to begin with. As mentioned, such an index already exists for `ssnum`. Thus, we can *cluster* this index according to [4] by performing:

```
CLUSTER Employee USING idx_ssnum;
```

Here we assumed the index on `ssnum` to be named `idx_ssnum`.

If we had to create such an index by ourselves, one could accomplish this by performing:

```
CREATE UNIQUE INDEX idx_ssnum ON Employee [USING BTREE] (ssnum);
```

The additional command in parentheses `USING BTREE` is optional, since it already is the default for Postgres (and Postgres only supports unique indexes using B-Trees) [3].

Now in order to create a clustered hash index, we first need to create the index itself following [1]:

```
CREATE INDEX idx_ssnum ON Employee USING HASH (ssnum);
```

We note that we now can no longer create a unique index, since it is not supported as mentioned before. Afterwards, we again can cluster this created index:

```
CLUSTER Employee USING idx_ssnum;
```

[1] Give the queries for creating a hash index *and* a B^+ tree index if both of them are supported.

b) Are clustering indexes on non-key attributes supported, e.g., on `name`? Show the query.

Yes, clustering indexes are also supported on non-key attributes. As mentioned earlier, we first need an index on name to cluster it:

```
CREATE INDEX idx_name on Employee (name);
```

Followed by that, we can now cluster this created index by performing:

```
CLUSTER Employee USING idx_name;
```

c) Is the clustering index dense or sparse?

In general, Postgres only supports dense indexes [5]. Hence, clustering indexes are dense as well. The only exception from this is the GIN index type, which is a somewhat sparse index.

d) How does the system deal with overflows in clustering indexes? How is the fill factor controlled?

[Your answer goes here ...]

e) Discuss any further characteristics of the system related to clustering indexes that are relevant to a database tuner.

[Your answer goes here ...]

3 Non-Clustering Indexes

Discuss how the system supports non-clustering indexes, in particular:

a) How do you create a combined, non-clustering index on `(dept,salary)`? Show the query.[1]

[Your answer goes here ...]

```
[Your SQL query goes here ...]
```

b) Can the system take advantage of covering indexes? What if the index covers the query, but the condition is not a prefix of the attribute sequence `(dept,salary)`?

[Your answer goes here ...]

c) Discuss any further characteristics of the system related to non-clustering indexes that are relevant to a database tuner.

[Your answer goes here ...]

4 Key Compression and Page Size

If your system supports B^+ trees, what kind of key compression (if any) is supported? How large is the default disk page? Can it be changed?

[Your answer goes here ...]

Time Spent on this Assignment

Time in hours per person:

- Florian Frauenschuh:
- Peter Lindner:
- Alexander Weilert:

References

[1] PostgreSQL. *Index Types*. Accessed: 2024-04-30. 2024. URL: https://www.postgresql.org/docs/14/indexes-types.html.

[2] PostgreSQL. *GIN and GiST Index Types*. Accessed: 2024-04-30. 2024. URL: https://www.postgresql.org/docs/9.1/textsearch-indexes.html.

[3] PostgreSQL. *Unique Indexes*. Accessed: 2024-04-30. 2024. URL: https://www.postgresql.org/docs/14/indexes-unique.html.

[4] PostgreSQL. *CLUSTER*. Accessed: 2024-04-30. 2024. URL: https://www.postgresql.org/docs/14/sql-cluster.html.

[5] user2993792. *PostgresSQL Indices - Are They Dense or Sparse?* Accessed: 2024-04-30. 2013. URL: https://stackoverflow.com/questions/19987786/postgresql-indices-are-they-dense-or-sparse.