Database Tuning – Assignment 3

Index Tuning

A2

Baumgartner Dominik, 0920177

Dafir Thomas Samy, 1331483

Schörgnhofer Kevin, 1421082

November 15, 2016

9.4/static/sql-cluster.html **Database system and version:** *PostgreSQL 9.6.0*

Question 1: Index Data Structures

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

B-tree, Hash, GiST, SP-GiST and GIN [1]

Question 2: Clustered Indexes

Discuss how the system supports clustered indexes, in particular:

2a) How do you create a clustered index on `ssnum`? Show the query.[2]

To create a clustered index in PostgreSQL you create an index first and cluster this index afterwards with the CLUSTER command. By default PostgreSQL creates a B^+-tree index. If you want a hash index, you have to specify it:[3] [4]

```
--creates a $B^+$-tree index and clusters it
CREATE INDEX ssnumE ON Employee (ssnum);
CLUSTER Emlpoyee USING ssnumE;
\\
--creates a hash index and clusters it
```

[1] PostgreSQL 9.6 Documentation, Chapter 11 Indexes, `https://www.postgresql.org/docs/9.6/static/indexes-types.html`

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

[3] PostgreSQL 9.6 Documentation, Chapter 11 Indexes, `https://www.postgresql.org/docs/9.6/static/indexes-types.html`

[4] PostgreSQL 9.6 Documentation, SQL Cluster, `https://www.postgresql.org/docs/9.6/static/sql-cluster.html`

```
CREATE INDEX ssnumE ON Employee USING HASH (ssnum);
CLUSTER Emlpoyee USING ssnumE;
```

2b) Are clustered indexes on non-key attributes supported, e.g., on `name`? Show the query.

Yes they are supported. In fact postgres supports clustering tables using any attribute used in an index. According to Postgres documentation there is no limitation to which index-type can be used.[5]

```
CLUSTER table USING index_name;
```

2c) Is the clustered index dense or sparse?

There is no answer to this question in the Postgres Documentation. So we derived a solution which seems logical: Every index which we create in Postgres is a non-clustered one, and hence dense[6]. With the $CLUSTER$ command we just cluster the existing data to match the index, which only rearranges the data entries and pointers and does not change the type (dense/spares) of the index. So the conclusion is, that the clustered indexes in Postgres are dense.

2d) How does the system deal with overflows in clustered indexes? How is the fill factor controlled?

The system does not resolve overflows at all. If overflows occur the table has to be manually clustered. Order preservation can be achieved by setting the fillfactor to a value below 100%. This enables the insertion of matching new data into pages inserted in the ordered structure. The table has to be reclustered.[7]

An index with a specified fillfactor is created as follows:[8]

```
CREATE UNIQUE INDEX index_title ON table (attribute) WITH (fillfactor = value);
```

2e) Discuss any further characteristics of the system related to clustered indexes that are relevant to a database tuner?

Tables using a clustered index have to be reclustered after every update to ensure they stay clustered. If a cluster is in progress an ACCESS EXCLUSIVE lock is acquired meaning no other read or write operation can be executed on the table. Since clustering a table according to an index sorts the table physically on the index attribute there can only be one clustered index per table. Furthermore the $CLUSTER$ instruction is not part of standard SQL but a specific feature of Postgresql.

---

[5] PostgreSQL 9.6 Documentation, SQL Cluster, `https://www.postgresql.org/docs/9.6/static/sql-cluster.html`

[6] DB-Tuning Lecture Notes, Index Tuning, Page 17

[7] PostgreSQL 9.6 Documentation, SQL Cluster, `https://www.postgresql.org/docs/9.6/static/sql-cluster.html`

[8] PostgreSQL 9.6 Documentation, SQL Create Index, `https://www.postgresql.org/docs/9.6/static/sql-createindex.html`

Clustering can be done using either a sequential table-scan (plus sorting) or an index scan. Both methods require a certain amount of diskspace: tablesize + indexsize for an index scan, double tablesize + indexsize for sequential table-scan. The method is selected automatically by the dbs whereas a sequential scan is often faster than an index scan. Due to the disk space requirements for a cluster it is advisable to set the *maintenanceworkmem* to a reasonable size.[9]

Question 3: Non-Clustered Indexes

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

3a) How do you create a non-clustered index on `(dept,salary)`? Show the query.

In Postgresql all indexes are non-clustered. Clustering has to be done manually. Postgres creates all indexes as B+ trees by default. If a hash index is desired it has to be specified in the query[10].

```
--creates a B+-tree index
CREATE INDEX deptSalary ON Employee (dept, salary);

--creates a hash index
CREATE INDEX deptSalary ON Employee USING HASH (dept, salary);
```

3b) 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)`?

The system takes advantage of covering indexes if the index covers the query and the condition is a prefix of the attribute sequence. If the condition is not a prefix of the attribute sequence the use of the index depends heavily on the type of query e.g. range queries hardly use the index.

3c) Discuss any further characteristics of the system related to non-clustered indexes that are relevant to a database tuner?

- A non-clustered index is the standard index type in Postgres
- There can be more than one non-clustered index per table
- Faster inserts, because the position where the data gets inserted does not matter
- $INCLUDE$ command is available, which enables us to include non-key parameters on the leaf level (not supported in Postgres)

[9] PostgreSQL 9.6 Documentation, SQL Cluster, `https://www.postgresql.org/docs/9.6/static/sql-cluster.html`

[10] PostgreSQL 9.6 Documentation, SQL Create Index, `https://www.postgresql.org/docs/9.6/static/sql-createindex.html`

Question 4: Key Compression and Page Size

If your system supports B^+-trees, what kind of key compression (if any) does it support? How large is the default disk page? Can it be changed?
PostgreSQL does not have key compression in B^+-trees.
PostgreSQL uses a fixed page size (commonly 8 kB), and does not allow tuples to span multiple pages, according to its documentation. A different page size can be selected when compiling the server.

Time in hours per person: **4**

Important: Reference your information sources!