Assignment 4

Index Tuning – Selection

Database Tuning

Start date: Nov 26, 2013
Due date: Dec 9, 2013, 23:59
Grading: 1 point

In this assignment you will experiment with indexes using PostgreSQL 9.2.

1. Download `http://www.cosy.sbg.ac.at/~augsten/teaching/dbt/dblp.zip` This archive contains two tab separated files (`publ.tsv` and `auth.tsv`) that store authors and their publications as found in the DBLP[1] bibliography. The imported tables have the following schemas:

   - `Auth(name(49),pubID(129))`
   - `Publ(pubID(129),type(13),title(700),booktitle(132),`
     `year(4),publisher(196))`

   You can assume that all attribute values are strings; the maximum string length is shown in brackets. `Publ.pubID` is a key.

2. Compare clustered B^+-tree, non-clustered B^+-tree, non-clustered hash index, and table scan (no index) for the following queries and measure the throughput:

```
SELECT * FROM Publ WHERE pubID = ...
SELECT * FROM Publ WHERE booktitle = ...
SELECT * FROM Publ WHERE year = ...
```

   (a) Explain your experimental setup, i.e., how did you send the queries to the database and how did you measure throughput?

   (b) Which conditions did you use for each of the query types (`pubID, booktitle, year`)? Use the same conditions for all index settings on a particular query type.

   (c) Give the throughput results and the query plan for each query type and each index setting.

   (d) Discuss your observations. Are the results expected? Why (not)?

[1] `http://www.informatik.uni-trier.de/~ley/db/`

Notes about the experimental setup

- To compute the throughput, measure the overall runtime for several runs of a query and divide the time by the number of queries. The overall runtime should be *more than one minute* to decrease the measurement error.
- When you repeat a query to measure the throughput, do not use the same condition in the `WHERE` clauses of the repeated queries. Instead, use *different* conditions for each call, for example, `year = 1980`, `year = 2001`, `year = 2004`, etc.
- Do *not* specify primary key, foreign key, or uniqueness constraints when you create the tables. PostgreSQL automatically creates an index to ensure uniqueness, which you want to avoid for some of the queries.
- To test the non-clustered indexes, cluster the table according to an attribute that is independent of the indexed attribute, e.g., cluster the table according to `title` for the condition on `year`.

Notes about PostgreSQL

- *Clustering indexes*: You first create an index, then you use the index to cluster the table (i.e., physically sort the table by the index attribute):

  ```
  CREATE INDEX year_idx ON publ(year);
  CLUSTER publ USING year_idx;
  ```

- *Query plan*: The command `EXPLAIN` shows the query plan without executing the query. The command `EXPLAIN ANALYSE` also executes the query. Example:

  ```
  EXPLAIN ANALYZE SELECT * FROM publ WHERE year='2006';
  ```

Please indicate the average time per group member that was spent solving this assignment. The time that you indicate will have *no* impact on your grade.