Database Tuning – Assignment 4

Index Tuning

A2
Baumgartner Dominik, 0920177
Dafir Thomas Samy, 1331483
Schörgnhofer Kevin, 1421082

November 29, 2016

Notes:

- Do not forget to run `ANALYZE tablename` after creating or changing a table.
- Use `EXPLAIN ANALYZE` for the query plans that you display in the report.

1 Experimental Setup

How do send the queries to the database? How do you measure the execution time for a sequence of queries?

2 Clustered B$^+$-Tree Index

Point Query Repeat the following query multiple times with different conditions for `pubID`.

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

Which conditions did you use?
Show the runtime results and compute the throughput.
Query plan (for one of the queries):

```
query plan
```

Multipoint Query – Low Selectivity Repeat the following query multiple times with different conditions for `booktitle`.

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

Which conditions did you use?
We used a equality condition where the index matches a given string from the booktitle row of the publ.tsv file, like 'Software Engineering (Workshops)'.
Show the runtime results and compute the throughput.
For the Clustered B$^+$-Tree Index we achieved a runtime from $9366ms$ to $10196ms$ with 719 searched values. This leads to a throughput of $70{,}5\frac{queries}{second}$ to $76{,}8\frac{queries}{second}$.

Query plan (for one of the queries):

```
EXPLAIN SELECT * FROM publ_cb WHERE booktitle = 'Software Engineering (Workshops)';

 Index Scan using booktitlecb on publ_cb  (cost=0.43..14.60 rows=181 width=112)
   Index Cond: ((booktitle)::text = 'Software Engineering (Workshops)'::text)
(2 rows)
```

Multipoint Query – High Selectivity Repeat the following query multiple times with different conditions for `year`.

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

Which conditions did you use?
Show the runtime results and compute the throughput.
Query plan (for one of the queries):

```
query plan
```

3 Non-Clustered B$^+$-Tree Index

Note: Make sure the data is not physically ordered by the indexed attributes due to the clustering index that you created before.

Point Query Repeat the following query multiple times with different conditions for `pubID`.

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

Which conditions did you use?
Show the runtime results and compute the throughput.
Query plan (for one of the queries):

```
query plan
```

Multipoint Query – Low Selectivity Repeat the following query multiple times with different conditions for `booktitle`.

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

Which conditions did you use?
We used a equality condition where the index matches a given string from the booktitle row of the publ.tsv file, like 'Software Engineering (Workshops)'.
Show the runtime results and compute the throughput.
For the Non-Clustered B$^+$-Tree Index we achieved a runtime from $9916ms$ to $11347ms$ with 719 searched values. This leads to a throughput of $63{,}4\frac{queries}{second}$ to $72{,}5\frac{queries}{second}$.

Query plan (for one of the queries):

```
EXPLAIN SELECT * FROM publ_b WHERE booktitle = 'Software Engineering (Workshops)';

 Index Scan using booktitleb on publ_b  (cost=0.43..577.16 rows=181 width=113)
   Index Cond: ((booktitle)::text = 'Software Engineering (Workshops)'::text)
(2 rows)
```

Multipoint Query – High Selectivity Repeat the following query multiple times with different conditions for `year`.

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

Which conditions did you use?

Show the runtime results and compute the throughput.
Query plan (for one of the queries):

```
query plan
```

4 Non-Clustered Hash Index

Note: Make sure the data is not physically ordered by the indexed attributes due to the clustering index that you created before.

Point Query Repeat the following query multiple times with different conditions for `pubID`.

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

Which conditions did you use?

Show the runtime results and compute the throughput.
Query plan (for one of the queries):

```
query plan
```

Multipoint Query – Low Selectivity Repeat the following query multiple times with different conditions for `booktitle`.

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

Which conditions did you use?
We used a equality condition where the index matches a given string from the booktitle row of the publ.tsv file, like 'Software Engineering (Workshops)'.

Show the runtime results and compute the throughput.
For the Non-Clustered Hash Index we achieved a runtime from $12447ms$ to $11347ms$ with 719 searched values. This leads to a throughput of $57{,}8\frac{queries}{second}$ to $76{,}9\frac{queries}{second}$.

Query plan (for one of the queries):

```
EXPLAIN SELECT * FROM publ_h WHERE booktitle = 'Software Engineering (Workshops)';

 Bitmap Heap Scan on publ_h  (cost=5.37..668.34 rows=177 width=112)
   Recheck Cond: ((booktitle)::text = 'Software Engineering (Workshops)'::text)
   ->  Bitmap Index Scan on booktitleh  (cost=0.00..5.33 rows=177 width=0)
         Index Cond: ((booktitle)::text = 'Software Engineering (Workshops)'::text)
(4 rows)
```

Multipoint Query – High Selectivity Repeat the following query multiple times with different conditions for `year`.

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

Which conditions did you use?
Show the runtime results and compute the throughput.
Query plan (for one of the queries):

```
query plan
```

5 Table Scan

Note: Make sure the data is not physically ordered by the indexed attributes due to the clustering index that you created before.

Point Query Repeat the following query multiple times with different conditions for `pubID`.

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

Which conditions did you use?
Show the runtime results and compute the throughput.
Query plan (for one of the queries):

```
query plan
```

Multipoint Query – Low Selectivity Repeat the following query multiple times with different conditions for `booktitle`.

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

Which conditions did you use?
We used a equality condition where the index matches a given string from the booktitle row of the publ.tsv file, like 'Software Engineering (Workshops)'.
Show the runtime results and compute the throughput.
For the Table Scan we achieved a runtime from 197998ms to 201809ms with 719 searched values. This leads to a throughput of 3,6$\frac{queries}{second}$.

Query plan (for one of the queries):

```
EXPLAIN SELECT * FROM publ_s WHERE booktitle = 'Software Engineering (Workshops)';

 Seq Scan on publ_s  (cost=0.00..37843.18 rows=179 width=112)
   Filter: ((booktitle)::text = 'Software Engineering (Workshops)'::text)
(2 rows)
```

Multipoint Query – High Selectivity Repeat the following query multiple times with different conditions for `year`.

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

Which conditions did you use?
Show the runtime results and compute the throughput.
Query plan (for one of the queries):

```
query plan
```

6 Discussion

Give the throughput of the query types and index types in queries/second.

	clustered	non-clust. B^+-tree	non-clust. hash	table scan
point (`pubID`)				
multipoint (`booktitle`)	73,14	69,37	69,73	3,6
multipoint (`year`)				

Discuss the runtime results for the different index types and the table scan. Are the results expected? Why / why not?

Time in hours per person: **XXX**

Important: Reference your information sources!