Assignment 4

Index Tuning – Selection

Database Tuning

Gruppe J
Brezovic Ivica, 11702570
Demir Cansu, 11700525
Mrazovic Mirna, 11700383

May 7, 2020

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.

Experimental Setup

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

Das Programm initialisiert sechs ResultSets: randomPubids, randomBooktitles, randomYears, randomList, randomList1 und randomList2 in der Query

```
SELECT ...  FROM "Publ" TABLESAMPLE BERNOULLI(5);
```

Anstelle von ... wird pubID, booktitle, name oder year eingesetzt.

Für jeden der Abfragetypen werden alle Indexes gelöscht und der entsprechende Index wird erstellt. Beispielsweise:

```
CREATE INDEX "publ_pubid_idx" ON "Publ" USING BTREE("pubID");
```

Die Erstellung der Tabellen und das ausführen der einzelnen Abfragen wurde mit dem Copymanager durchgeführt, da es sich um die schnellste Verbindung handelt.

Die Zeitmessung haben wir mit System.nanotime() durchgeführt. Es wurde ein Zeitstempel vor der Abfrage positioniert und einer danach. Danach wurde die Beginnzeit von der Endzeit subtrahiert und das Ergebnis in Sekunden umgewandelt.

Jede Query wird mit con.createStatement().execute("...") an den Server gesendet, und ein counter (executedQueries) zählt die Anzahl der ausgeführten Queries.

Clustering 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?

Die WHERE Bedingung wird zufällig aus dem ResultSet randomPubid ausgewählt, welches beim Start des Programms mit folgender Query initialisiert wird:

```
SELECT "pubID" FROM "Publ" TABLESAMPLE BERNOULLI(5);
```

Point Query on server:
Show the runtime results and compute the throughput.

Elapsed time: 60.00128 seconds
Throughput [1/s]: 184

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 1--
Index Scan using "publ_PUBID_idx" on "Publ"  (cost=0.43..8.45 rows=1 width=112)
                                             (actual time=0.055..0.056 rows=1 loops=1)
  Index Cond: (("pubID")::text = 'conf/esann/FyfeL07'::text)
Planning time: 0.304 ms
Execution time: 0.104 ms
```

Point Query on localhost:
Show the runtime results and compute the throughput.

Elapsed time: 60.000515 seconds
Throughput [1/s]: 1889

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 1--
Index Scan using "publ_PUBID_idx" on "Publ"  (cost=0.43..8.45 rows=1 width=112)
                                             (actual time=0.142..0.144 rows=1 loops=1)
  Index Cond: (("pubID")::text = 'journals/dm/ChenYY01'::text)
Planning Time: 0.544 ms
Execution Time: 0.181 ms
```

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

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

```
SELECT * FROM Publ WHERE pubID IN (...)
```

Which conditions did you use?

Die WHERE Bedingung wird zufällig aus dem ResultSet randomBooktitles ausgewählt, welches beim Start des Programms mit folgender Query initialisiert wird:

```
SELECT "booktitle" FROM "Publ" TABLESAMPLE BERNOULLI(5);
```

Multipoint Query on server:
Show the runtime results and compute the throughput.

Elapsed time: 60.003178 seconds
Throughput [1/s]: 207

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 2--
Index Scan using "publ_BOOKTITLE_idx" on "Publ"  (cost=0.43..8.45 rows=1 width=112)
                                                 (actual time=0.031..0.031 rows=0 loops=1)
  Index Cond: (("pubID")::text = 'Internet der Dinge'::text)
Planning time: 0.360 ms
Execution time: 0.055 ms
```

Multipoint Query on localhost:
Show the runtime results and compute the throughput.

Elapsed time: 60.000153 seconds
Throughput [1/s]: 2945

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 2--
Index Scan using "publ_BOOKTITLE_idx" on "Publ"  (cost=0.43..8.45 rows=1 width=112)
                                                 (actual time=0.095..0.095 rows=0 loops=1)
  Index Cond: (("pubID")::text = 'Internet der Dinge'::text)
Planning Time: 0.446 ms
Execution Time: 0.114 ms
```

Multipoint Query IN-Predicate on server:
Show the runtime results and compute the throughput.

Elapsed time: 60.00348 seconds
Throughput [1/s]: 172

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 3--
Index Scan using "publ_LIST_idx" on "Publ"  (cost=0.43..17.34 rows=3 width=112)
                                            (actual time=0.033..0.047 rows=3 loops=1)
  Index Cond: (("pubID")::text = ANY ('{series/vdi/PflaumH07,series/vdi/
               LiekenbrockE07,series/vdi/SchenkR07}'::text[]))
Planning time: 0.465 ms
Execution time: 0.091 ms
```

Multipoint Query IN-Predicate on localhost:
Show the runtime results and compute the throughput.

Elapsed time: 60.000095 seconds
Throughput [1/s]: 2345

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 3--
Index Scan using "publ_LIST_idx" on "Publ"  (cost=0.43..17.34 rows=3 width=112)
                                            (actual time=0.166..0.215 rows=3 loops=1)
  Index Cond: (("pubID")::text = ANY ('{series/vdi/PaterS07,series/vdi/
               LiekenbrockE07,series/vdi/Bock07}'::text[]))
Planning Time: 2.527 ms
Execution Time: 0.245 ms
```

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?

Die WHERE Bedingung wird zufällig aus dem ResultSet randomYears ausgewählt, welches beim Start des Programms mit folgender Query initialisiert wird:

```
SELECT "year" FROM "Publ" TABLESAMPLE BERNOULLI(5);
```

Multipoint Query on server:
Show the runtime results and compute the throughput.

Elapsed time: 60.01319 seconds
Throughput [1/s]: 97

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 4--
Index Scan using "publ_YEAR_idx" on "Publ"  (cost=0.43..8.45 rows=1 width=112)
                                            (actual time=0.013..0.013 rows=0 loops=1)
  Index Cond: (("pubID")::text = '2007'::text)
Planning time: 0.076 ms
Execution time: 0.031 ms
```

Multipoint Query on localhost:
Show the runtime results and compute the throughput.

Elapsed time: 60.000145 seconds
Throughput [1/s]: 1631

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 4--
Index Scan using "publ_YEAR_idx" on "Publ"  (cost=0.43..8.45 rows=1 width=112)
                                            (actual time=0.058..0.058 rows=0 loops=1)
  Index Cond: (("pubID")::text = '2006'::text)
Planning Time: 0.261 ms
Execution Time: 0.084 ms
```

Non-Clustering 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?

Die WHERE Bedingung wird zufällig aus dem ResultSet randomPubids ausgewählt, welches beim Start des Programms mit folgender Query initialisiert wird:

```
SELECT "pubID" FROM "Publ" TABLESAMPLE BERNOULLI(5);
```

Point Query on server:
Show the runtime results and compute the throughput.

Elapsed time: 60.00515 seconds
Throughput [1/s]: 187

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 1--
Index Scan using "publ_PUBID_idx" on "Publ"  (cost=0.43..8.45 rows=1 width=1548)
                                             (actual time=0.061..0.062 rows=1 loops=1)
  Index Cond: (("pubID")::text = 'journals/bioinformatics/BatemanV06'::text)
Planning time: 0.292 ms
Execution time: 0.086 ms
```

Point Query on localhost:
Show the runtime results and compute the throughput.

Elapsed time: 60.00043 seconds
Throughput [1/s]: 2035

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 1--
Index Scan using "publ_PUBID_idx" on "Publ" (cost=0.43..8.45 rows=1 width=1548)
                                            (actual time=0.329..0.333 rows=1 loops=1)
  Index Cond: (("pubID")::text = 'journals/cj/FraenkelK90'::text)
Planning Time: 3.279 ms
Execution Time: 0.376 ms
```

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

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

SELECT * FROM Publ WHERE pubID IN (...)
```

Which conditions did you use?

Die WHERE Bedingung wird zufällig aus dem ResultSet randomBooktitles ausgewählt, welches beim Start des Programms mit folgender Query initialisiert wird:

```
SELECT "booktitle" FROM "Publ" TABLESAMPLE BERNOULLI(5);
```

Multipoint Query on server:
Show the runtime results and compute the throughput.

Elapsed time: 60.000256 seconds
Throughput [1/s]: 174

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 2--
Index Scan using "publ_BOOKTITLE_idx" on "Publ" (cost=0.43..8.45 rows=1 width=1548)
                                                (actual time=0.038..0.038 rows=0 loops=1)
  Index Cond: (("pubID")::text = 'Internet der Dinge'::text)
Planning time: 0.207 ms
Execution time: 0.062 ms
```

Multipoint Query on localhost:
Show the runtime results and compute the throughput.

Elapsed time: 60.000046 seconds
Throughput [1/s]: 2109

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 2--
Index Scan using "publ_BOOKTITLE_idx" on "Publ" (cost=0.43..8.45 rows=1 width=112)
                                                (actual time=0.192..0.193 rows=0 loops=1)
  Index Cond: (("pubID")::text = 'Internet der Dinge'::text)
Planning Time: 3.606 ms
Execution Time: 0.235 ms
```

Multipoint Query IN-Predicate on server:
Show the runtime results and compute the throughput.

Elapsed time: 60.21634 seconds
Throughput [1/s]: 164

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 3--
Index Scan using "publ_LIST_idx" on "Publ" (cost=0.43..18.02 rows=3 width=112)
                                           (actual time=0.038..0.055 rows=3 loops=1)
  Index Cond: (("pubID")::text = ANY ('{series/vdi/PflaumH07,series/vdi/
               LiekenbrockE07,series/vdi/SchenkR07}'::text[]))
Planning time: 0.298 ms
Execution time: 0.086 ms
```

Multipoint Query IN-Predicate on localhost:
Show the runtime results and compute the throughput.

Elapsed time: 60.000473 seconds
Throughput [1/s]: 1986

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 3--
Index Scan using "publ_LIST_idx" on "Publ" (cost=0.43..18.07 rows=3 width=112)
                                           (actual time=0.240..0.337 rows=3 loops=1)
  Index Cond: (("pubID")::text = ANY ('{series/vdi/PaterS07,series/vdi/
               LiekenbrockE07,series/vdi/Bock07}'::text[]))
Planning Time: 4.468 ms
Execution Time: 0.382 ms
```

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?

Die WHERE Bedingung wird zufällig aus dem ResultSet randomYears ausgewählt, welches beim Start des Programms mit folgender Query initialisiert wird:

```
SELECT "year" FROM "Publ" TABLESAMPLE BERNOULLI(5);
```

Multipoint Query on server:
Show the runtime results and compute the throughput.

Elapsed time: 60.00979 seconds
Throughput [1/s]: 97

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 4--
Index Scan using "publ_YEAR_idx" on "Publ" (cost=0.43..8.45 rows=1 width=112)
                                           (actual time=0.018..0.019 rows=0 loops=1)
  Index Cond: (("pubID")::text = '2007'::text)
Planning time: 0.093 ms
Execution time: 0.036 ms
```

Multipoint Query on localhost:
Show the runtime results and compute the throughput.

Elapsed time: 60.000072 seconds
Throughput [1/s]: 1263

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 4--
Index Scan using "publ_YEAR_idx" on "Publ" (cost=0.43..8.45 rows=1 width=112)
                                           (actual time=0.046..0.047 rows=0 loops=1)
  Index Cond: (("pubID")::text = '2006'::text)
Planning Time: 0.264 ms
Execution Time: 0.072 ms
```

Non-Clustering 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?

Die WHERE Bedingung wird zufällig aus dem ResultSet randomPubid ausgewählt, welches beim Start des Programms mit folgender Query initialisiert wird:

```
SELECT "pubID" FROM "Publ" TABLESAMPLE BERNOULLI(5);
```

Point Query on server:
Show the runtime results and compute the throughput.

Elapsed time: 60.00068 seconds
Throughput [1/s]: 174

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 1--
Index Scan using "publ_PUBID_idx" on "Publ" (cost=0.00..8.02 rows=1 width=1548)
                                            (actual time=0.049..0.051 rows=1 loops=1)
```

```
  Index Cond: (("pubID")::text = 'conf/icalt/LundNSV05'::text)
Planning time: 0.176 ms
Execution time: 0.078 ms
```

Point Query on localhost:
Show the runtime results and compute the throughput.

Elapsed time: 60.00053 seconds
Throughput [1/s]: 2321

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 1--
Index Scan using "publ_PUBID_idx" on "Publ" (cost=0.00..8.02 rows=1 width=1548)
                                            (actual time=0.117..0.119 rows=1 loops=1)
  Index Cond: (("pubID")::text = 'conf/nma/Chryssoverghi06'::text)
Planning Time: 3.495 ms
Execution Time: 0.153 ms
```

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

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

```
SELECT * FROM Publ WHERE pubID IN (...)
```

Which conditions did you use?

Die WHERE Bedingung wird zufällig aus dem ResultSet randomBooktitles ausgewählt, welches beim Start des Programms mit folgender Query initialisiert wird:

```
SELECT "booktitle" FROM "Publ" TABLESAMPLE BERNOULLI(5);
```

Multipoint Query on server:
Show the runtime results and compute the throughput.

Elapsed time: 60.005013 seconds
Throughput [1/s]: 189

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 2--
Index Scan using "publ_BOOKTITLE_idx" on "Publ" (cost=0.00..8.02 rows=1 width=112)
                                                (actual time=0.012..0.012 rows=0 loops=1)
  Index Cond: (("pubID")::text = 'Internet der Dinge'::text)
Planning time: 0.172 ms
Execution time: 0.034 ms
```

Multipoint Query on localhost:
Show the runtime results and compute the throughput.

Elapsed time: 60.00004 seconds
Throughput [1/s]: 2553

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 2--
Index Scan using "publ_BOOKTITLE_idx" on "Publ" (cost=0.00..8.02 rows=1 width=112)
                                                (actual time=0.014..0.014 rows=0 loops=1)
  Index Cond: (("pubID")::text = 'Internet der Dinge'::text)
Planning Time: 2.323 ms
Execution Time: 0.035 ms
```

Multipoint Query IN-Predicate on server:
Show the runtime results and compute the throughput.

Elapsed time: 60.006603 seconds
Throughput [1/s]: 192

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 3--
Index Scan using "Publ_pkey" on "Publ" (cost=0.43..18.01 rows=3 width=112)
                                       (actual time=0.863..0.891 rows=3 loops=1)
  Index Cond: (("pubID")::text = ANY ('{series/vdi/PflaumH07,series/vdi/
               LiekenbrockE07,series/vdi/SchenkR07}'::text[]))
Planning time: 0.188 ms
Execution time: 0.926 ms
```

Multipoint Query IN-Predicate on localhost:
Show the runtime results and compute the throughput.

Elapsed time: 60.000454 seconds
Throughput [1/s]: 2312

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 3--
Index Scan using "Publ_pkey" on "Publ" (cost=0.43..18.10 rows=3 width=112)
                                       (actual time=0.147..0.168 rows=3 loops=1)
  Index Cond: (("pubID")::text = ANY ('{series/vdi/PaterS07,series/vdi/
               LiekenbrockE07,series/vdi/Bock07}'::text[]))
Planning Time: 1.739 ms
Execution Time: 0.182 ms
```

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?

Die WHERE Bedingung wird zufällig aus dem ResultSet randomYears ausgewählt, welches beim Start des Programms mit folgender Query initialisiert wird:

```
SELECT "year" FROM "Publ" TABLESAMPLE BERNOULLI(5);
```

Multipoint Query on server:
Show the runtime results and compute the throughput.

Elapsed time: 60.002354 seconds
Throughput [1/s]: 93

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 4--
Index Scan using "publ_YEAR_idx" on "Publ" (cost=0.00..8.02 rows=1 width=112)
                                           (actual time=0.007..0.008 rows=0 loops=1)
  Index Cond: (("pubID")::text = '2007'::text)
Planning time: 0.069 ms
Execution time: 0.024 ms
```

Multipoint Query on localhost:
Show the runtime results and compute the throughput.

Elapsed time: 60.000504 seconds
Throughput [1/s]: 1333

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 4--
Index Scan using "publ_YEAR_idx" on "Publ" (cost=0.00..8.02 rows=1 width=112)
                                           (actual time=0.016..0.016 rows=0 loops=1)
  Index Cond: (("pubID")::text = '2006'::text)
Planning Time: 0.326 ms
Execution Time: 0.049 ms
```

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 = ...

SELECT * FROM Publ WHERE pubID IN (...)
```

Which conditions did you use?

Die WHERE Bedingung wird zufällig aus dem ResultSet randomPubids ausgewählt, welches beim Start des Programms mit folgender Query initialisiert wird:

```
SELECT "pubID" FROM "Publ" TABLESAMPLE BERNOULLI(5);
```

Point Query on server:
Show the runtime results and compute the throughput.

Elapsed time: 60.00076 seconds
Throughput [1/s]: 191

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 1--
Index Scan using "Publ_pkey" on "Publ"  (cost=0.42..8.44 rows=1 width=1548)
                                        (actual time=0.078..0.079 rows=1 loops=1)
  Index Cond: (("pubID")::text = 'journals/jlp/VerdejoM06'::text)
Planning time: 0.457 ms
Execution time: 0.150 ms
```

Point Query on localhost:
Show the runtime results and compute the throughput.

Elapsed time: 60.00047 seconds
Throughput [1/s]: 2106

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 1--
Index Scan using "Publ_pkey" on "Publ" (cost=0.42..8.44 rows=1 width=1548)
                                       (actual time=0.073..0.074 rows=1 loops=1)
  Index Cond: (("pubID")::text = 'conf/ijcai/Oleson77'::text)
Planning Time: 0.138 ms
Execution Time: 0.093 ms
```

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

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

SELECT * FROM Publ WHERE pubID IN (...)
```

Which conditions did you use?

Die WHERE Bedingung wird zufällig aus dem ResultSet randomBooktitles ausgewählt, welches beim Start des Programms mit folgender Query initialisiert wird:

```
SELECT "booktitle" FROM "Publ" TABLESAMPLE BERNOULLI(5);
```

Multipoint Query on server:
Show the runtime results and compute the throughput.

Elapsed time: 60.003128 seconds
Throughput [1/s]: 163

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 2--
Index Scan using "Publ_pkey" on "Publ" (cost=0.42..8.44 rows=1 width=1548)
                                       (actual time=0.022..0.022 rows=0 loops=1)
  Index Cond: (("pubID")::text = 'Internet der Dinge'::text)
Planning time: 0.072 ms
Execution time: 0.043 ms
```

Multipoint Query on localhost:
Show the runtime results and compute the throughput.

Elapsed time: 60.000015 seconds
Throughput [1/s]: 3480

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 2--
Index Scan using "Publ_pkey" on "Publ" (cost=0.43..8.45 rows=1 width=112)
                                       (actual time=0.037..0.037 rows=0 loops=1)
  Index Cond: (("pubID")::text = 'Internet der Dinge'::text)
Planning Time: 0.107 ms
Execution Time: 0.057 ms
```

Multipoint Query IN-Predicate on server:
Show the runtime results and compute the throughput.

Elapsed time: 60.00082 seconds
Throughput [1/s]: 174

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 3--
Index Scan using "Publ_pkey" on "Publ" (cost=0.43..18.01 rows=3 width=112)
                                       (actual time=0.023..0.042 rows=3 loops=1)
  Index Cond: (("pubID")::text = ANY ('{series/vdi/PflaumH07,series/vdi/
               LiekenbrockE07,series/vdi/SchenkR07}'::text[]))
Planning time: 0.091 ms
Execution time: 0.065 ms
```

Multipoint Query IN-Predicate on localhost:
Show the runtime results and compute the throughput.

Elapsed time: 60.00005 seconds
Throughput [1/s]: 2303

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 3--
Index Scan using "Publ_pkey" on "Publ" (cost=0.43..18.10 rows=3 width=112)
                                       (actual time=0.084..0.118 rows=3 loops=1)
  Index Cond: (("pubID")::text = ANY ('{series/vdi/PaterS07,series/vdi/
              LiekenbrockE07,series/vdi/Bock07}'::text[]))
Planning Time: 0.109 ms
Execution Time: 0.132 ms
```

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?

Die WHERE Bedingung wird zufällig aus dem ResultSet randomYears ausgewählt, welches beim Start des Programms mit folgender Query initialisiert wird:

```
SELECT "year" FROM "Publ" TABLESAMPLE BERNOULLI(5);
```

Multipoint Query on server:
Show the runtime results and compute the throughput.

Elapsed time: 60.00195 seconds
Throughput [1/s]: 95

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 4--
Index Scan using "Publ_pkey" on "Publ"  (cost=0.43..8.45 rows=1 width=112)
                                        (actual time=0.013..0.013 rows=0 loops=1)
  Index Cond: (("pubID")::text = '2007'::text)
Planning time: 0.049 ms
Execution time: 0.029 ms
```

Multipoint Query on localhost:
Show the runtime results and compute the throughput.

Elapsed time: 60.00008 seconds
Throughput [1/s]: 2000

Query plan (for one of the queries):

```
--EXPLAIN ANALYZE QUERY 4--
Index Scan using "Publ_pkey" on "Publ"  (cost=0.43..8.45 rows=1 width=112)
                                        (actual time=0.045..0.045 rows=0 loops=1)
  Index Cond: (("pubID")::text = '2006'::text)
Planning Time: 0.116 ms
Execution Time: 0.067 ms
```

Discussion

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

Server

	clustering B^+ tree	non-clust. B^+ tree	non-clust. hash	table scan
point (`pubID`)	184	187	174	191
multipoint (`booktitle`)	207	174	189	163
multipoint-IN (`pubID`)	172	164	192	174
multipoint (`year`)	97	97	93	95

Localhost

	clustering B^+ tree	non-clust. B^+ tree	non-clust. hash	table scan
point (`pubID`)	4733	5531	6083	5372
multipoint (`booktitle`)	6224	6419	6710	7103
multipoint-IN (`pubID`)	3563	4719	4062	5630
multipoint (`year`)	3091	3097	3419	3924

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

Für alle Ergebnisse kann man sagen, dass der Throughput pro Sekunde bei Localhost deutlich höher ist als beim Server. Da sich der Localhost (wie der Name schon sagt) lokal auf unserem Gerät befindet, schaffen wir hier ein etwa 10 mal besseres Ergebnis als beim Server.

Für die meisten Abfragen kann man gut sehen, dass der Table-Scan die langsamste Methode ist. Das liegt an der Art des Table-Scan, welcher grundsätzlich immer durch die ganze Tabelle scannt.

Der clustering-B+Tree ist grundsätzlich etwas schneller als der non-clustering-B+Tree. Beim clustering-B+Tree werden die Daten in index ordner gereiht, während das beim non-clustering-B+Tree nicht der Fall ist. Der non-clustering-B+Tree kann mehrmals in der Tabelle verwendet werden und eignet sich dafür eher für insert und update Operationen.

Bei Point Queries ist die hashbasierte Zugriffsmethode am schnellsten, aber nicht viel schneller. Im Vergleich zu einer Tabellenabfrage ist der hashbasierte Zugriff jedoch schneller. Dies ist nicht überraschend, da alle anderen Methoden das Lesen der gesamten Tabelle vermeiden.

Bei einer Hashtabelle kann man mit einem Primärschlüssel die Werte aus der Tabelle bekommen. Da das eine Laufzeit von O(1) ergibt, ist diese Zugriffsmethode etwas schneller als O(log(n)), welche wir bei einem B+ Baum brauchen

Time Spent on this Assignment

Time in hours per person: **9**

References

Important: Reference your information sources!
Remove this section if you use footnotes to reference your information sources.
