Duc Tran

S3752703@student.rmit.edu.au

COSC2406/2407 – DATABASE SYSTEMS  
ASSIGNMENT #2

Experimenting with secondary index in derby and mongoDB and implent B+ tree java

Overview

The experiment requires testing four quires on a Derby and Mongo database. The queries on these two types of databases would be tested with and without a secondary index. The structure of the database is based on the tester assignment 1, not based on the sample answer provided.

Part 1 – Derby

The four queries test on Derby DB were:

1. SELECT HOURLY\_COUNTS FROM COUNTS WHERE ID IN (SELECT ID

FROM MAIN

WHERE SENSOR\_ID >30);

1. SELECT HOURLY\_COUNT FROM COUNTS WHERE ID IN (SELECT ID

FROM MAIN

WHERE SENSOR\_ID IN (

SELECT SENSOR\_ID

FROM SENSOR\_DETAILS

WHERE SENSOR\_NAME > "A")) fetch first 5 rows only;

1. SELECT COUNT(ID) FROM COUNTS WHERE YEAR0 = 2009;
2. SELECT COUNT(ID) FROM COUNTS WHERE HOURLY\_COUNTS = 300;

And to create the secondary index in the database

CREATE INDEX SENSOR\_NAME\_ASC ON SENSOR\_DETAILS(SENSOR\_NAME ASC);

CREATE INDEX SENSOR\_ID\_ASC ON MAIN(SENSOR\_ID ASC);

The sensor ID in the main table and sensor name in the sensor details is chosen because that is the field that would be frequently searched, that or sensor name. Also, the choice to do it in the main table is also to decrease the number of rows that Derby must scan and therefore increase the query performance.

As shown in the table 1.1 and 1.2, the performance increase for query 2 is significant, although query 1 does not show any improvement.

As my secondary is sorted in ascending order on the sensor ID and sensor name field, it is expected that queries 3 and 4 do not see any performance because it does not use secondary index.

**1.1 No Index Derby**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Without secondary index | | | | | |
|  | Query1 | Query2 | Query3 | Query4 |  |
| Attemp1 | 5818 | 72219 | 729 | 390 | ms |
| Attemp2 | 6220 | 72214 | 829 | 414 | ms |
| Attemp3 | 6176 | 73192 | 749 | 408 | ms |

**1.2 Indexed Derby**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| With secondary index | | | | | |
|  | Query1 | Query2 | Query3 | Query4 |  |
| Attemp1 | 210 | 50767 | 887 | 439 | ms |
| Attemp2 | 210 | 50915 | 687 | 455 | ms |
| Attemp3 | 209 | 50750 | 770 | 404 | ms |

Part 2 Mongo

Conducting the same experiment as part 1 with slight changes to query 1 and query 2 to decrease the performance on these 2 queries (otherwise the execution time is always 0)

The queries:

1. db.Pedes\_count.find({$and:[{Sensor\_ID:{$gt:30}},{"Padestrian\_item.Month":"July"},{"Padestrian\_item.Hourly\_Counts":{$gt:10000}}]},{"Padestrian\_item.ID":1,"Padestrian\_item.Hourly\_Counts":1}).pretty().explain("executionStats")
2. db.Pedes\_count.find({$and:[{Sensor\_Name:{$gt:"A"}},{"Padestrian\_item.Month":"July"},{"Padestrian\_item.Hourly\_Counts":{$gt:10000}}]},{"Padestrian\_item.ID":1,"Padestrian\_item.Hourly\_Counts":1}).pretty().explain("executionStats")
3. db.Pedes\_count.explain("executionStats").distinct("Padestrian\_item.ID",{"Padestrian\_item.Year":2009})
4. db.Pedes\_count.explain("executionStats").distinct("Padestrian\_item.ID",{"Padestrian\_item.Hourly\_Counts":300})

And to create secondary index

db.Pedes\_count.createIndex({Sensor\_ID:1})

db.Pedes\_count.createIndex({Sensor\_Name:1})

The result the presented in table 2.1 and 2.3 shown that Mongo is way faster than Derby even without indexing. This may be the result of an NO-SQL database compared to Derby relational database where tables must be joined before getting the information.

* 1. No Index Mongo

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Query1 | Query2 | Query3 | Query4 |  |
| Attemp1 | 326 | 288 | 20 | 20 | ms |
| Attemp2 | 289 | 286 | 21 | 20 | ms |
| Attemp3 | 292 | 287 | 20 | 20 | ms |

* 1. Indexed Mongo

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| With secondary index | | | | | |
|  | Query1 | Query2 | Query3 | Query4 |  |
| Attemp1 | 0 | 0 | 20 | 20 | ms |
| Attemp2 | 0 | 0 | 21 | 20 | ms |
| Attemp3 | 0 | 0 | 20 | 20 | ms |

Link to git: https://github.com/s3752703/COSC2406-ASSIGNMENT2