December 06, 2018
Arnab Chakrabarti, Rihan Hai

Implementation of Databases (WS 18/19)

Exercise 5

Due until December 18, 2018, 10am.
Please submit your solution *in a single PDF file* before the deadline to the L^{2}P system!
Please submit solutions in groups of three students.

Exercise 5.1 (Database Tuning) (10 pts)

Consider the following schema of a nation-wide university database:

- $student(\underline{sid}, sname, sex, age, year, gpa)$: students with name, sex, age, year and their GPA
- $dept(\underline{did}, dname, numphds)$: departments with name and the number of phds in this department
- $prof(\underline{pid}, pname, did)$: professors with name and their department
- $course(\underline{cid}, cname, did)$: course names and its department
- $major(\underline{did, sid})$: the major departments of a student (a student might have more than one major)
- $enroll(\underline{sid, cid}, grade)$: the enrollment relationship between students and courses

A SQL script with CREATE TABLE statements is provided as attachment to this task in L2P. Note that the SQL script does not include any constraints (no primary key or foreign key definitions). These queries should be considered (two SQL statements for each query are given below):

1. For each Computer Science course (a course from a department starting with 'c'), get the course name and the average gpa of the students enrolled in the course.
2. Get the names and departments of all courses with more than 40 students enrolled in them.
3. Get the name(s) of the student(s) enrolled in the most classes.

Query1A

```
SELECT c.cname, AVG(s.gpa)
FROM course c, enroll e, student s, dept d
WHERE e.sid = s.sid AND c.did=d.did AND
d.dname LIKE 'c%' AND
e.cid=c.cid
GROUP BY c.cid, c.cname;
```

Query1B

```
SELECT c2.cname, AVG(s.gpa)
FROM enroll e, student s, course c2
WHERE e.sid = s.sid AND
e.cid IN (SELECT cid FROM course c, dept d
WHERE c.did=d.did AND d.dname LIKE 'c%')
GROUP BY c2.cid, c2.cname;
```

Query2A

```
SELECT t.cname, t.dname
FROM (SELECT c.cname, d.dname
FROM enroll e, course c, dept d
WHERE e.cid = c.cid AND d.did=c.did) AS t
GROUP BY t.cname, t.dname
HAVING COUNT(*) > 40;
```

Query2B

```
SELECT c.cname, d.dname
FROM course c, dept d,
(SELECT e.cid, COUNT(*) AS count
FROM enroll e GROUP BY e.cid) AS t
WHERE c.cid=t.cid AND c.did=d.did
AND t.count > 40;
```

Query3A

```
SELECT  s.sname
FROM    student s
WHERE s.sid IN (SELECT e.sid
FROM enroll e GROUP BY e.sid
HAVING COUNT(*)=(SELECT MAX(count)
FROM(SELECT COUNT(*) as count
FROM enroll e2
GROUP BY e2.sid) AS foo));
```

Query3B

```
SELECT s.sname FROM student s, enroll e
WHERE  s.sid=e.sid
GROUP BY s.sid, s.sname
HAVING COUNT(*) >= ALL
(SELECT COUNT(*)
FROM student s2, enroll e2
WHERE s2.sid=e2.sid
GROUP BY s2.sid);
```

1. Draw the estimated execution plan for each query (i.e., an operator tree of algebra expressions and segment/index scans as it is provided by using the EXPLAIN statement in PostgreSQL). Estimate which query plan will perform better (Variant A or B) and argue why. You can use your favorite DBMS for this task, but we recommend PostgreSQL. **(4 pts)**

2. **Performance Tuning** **(6 pts)**
   The database is heavily used with the queries above and some additional queries. Furthermore, updates, inserts and deletes are also performed on the database. This generates a huge workload and the performance is going down. Your task as a database administrator is to increase the performance for the given workload.

   Note: Include the updated SQL script of your submission. Make sure that the modified query really returns the same result as the original query (for all possible DB instances, not just for the provided sample databases).Please provide also Information about the result of genWorkload on your system and describe your system briefly (CPU, RAM, HDD or SSD).

   Consider the following informations for the Tuning Task:

   - The data is managed in a PostgreSQL server 10. The server has 8 GB of RAM and a CPU with 6 cores. There are also other applications running on this server, so you may not request all resources for the database server. The data is stored on classical hard disk (no SSD). You may change the configuration of your server (i.e., the file postgresql.conf); if you do so, explain your changes briefly in the PDF document, and include the modified postgresql.conf **in the submitted ZIP file.**
   - The database dump is available in L2P under Shared Documents. For testing, there are also smaller database dumps available. Note that you should first load the schema as the dump contains only data. You can use pgrestore (or the GUI pgadmin) to restore the database:

     ```
     pg_restore --host localhost --port 5432 --username "postgres" --dbname "db"
     --no-password  --section data "file.backup"
     ```

   - The workload is generated by the stored procedure $genWorkload(N, Q)$. N is the number of statements which will be generated, and Q is the percentage of queries in the workload. In general, we assume that there are about 80% queries and 20% inserts, updates, and deletes. The genWorkload function will return the required in an exception message. The exception is raised to abort the transaction and to avoid a change in the database. Thus, you can repeatedly run the function without changing the database state. **The goal is to minimize the time required for the query** `SELECT genWorkload(100,80).` We recommend to start with the tiny database and smaller values for N, because without any optimizations or indexes, the workload might take hours or days.

- Please note that you should not change or remove existing data in the dump nor make any assumptions about specific characteristics of the data (e.g., assuming that all strings have a fixed length as in the provided DB dumps).

Exercise 5.2 (Query Optimization in MongoDB) (15 pts)

1. Load the previous university data into MongoDB

   (a) A SQL script with CREATE TABLE **flat** (*createFlatTbl.sql*) is given on L2P, which stores the joined results of the four tables *student*, *enroll*, *dept*, *course* as Query1A from in Task 5.1.1 but without checking the department name. Execute the script; export the data in the table **flat** to a CSV file (e.g., flat.csv); create a new collection in MongoDB (e.g., also name it as flat) and load the data from flat.csv to this collection. **(1 pts)**

   (b) Write a MongoDB query *Q5* on the new collection **flat** for the below requirement. Provide the query execution time. (Note in the new collection **flat** it already contains the joined results from original four tables.) **(3 pts)**

   (Q5) For each Computer Science course (a course from a department starting with c), get the course id and the average gpa of the students enrolled in the course.

2. Improve the MongoDB query performance as below:

   - Redesign the schema of the collection **flat** according the query *Q5* such that the execution time of *Q5* is significantly reduced. Create a new collection with your redesigned schema (e.g., name the new collection as **nested**). Transform the data from the collection **flat** to the collection **nested**. You can use any kind of tool or language (e.g., Java, Python, Apache Spark, ...) for this data transformation task. Assume in this scenario data redundancy is not a problem. **(7 pts)**
   - Write a new MongoDB query *Q6* on the new collection **nested** which satisfies the same requirement in 5.2.1(b) as *Q5*. Provide the query execution time of *Q6*. **(2 pts)**

3. (Continue with first task 5.2.1). Build index(es) on the collection **flat** to improve the performance of *Q5*. Provide the new query execution time of *Q5*. Note MongoDB will automatically create an index on *_id*. **(2 pts)**

Exercise 5.3 (Short Questions) (5 pts)

Answer the following questions:

1. Explain the concept of "Polyglot Persistence" in NoSQL systems? (2 pt.)
2. Consider a document-oriented database with documents in the following form: (3 pt.)

   { type: "invoice", invoiceid: 5748, total : 25.0, billingcountry : "Germany" }

   { type : "invoiceline", lineid : 4378, invoiceid : 12234, trackid : 23, price : 0.99, quantity: 1 }

{ type : “track”, trackid : 1234, name : “Let There Be Rock”, albumid : 45 }

Provide the corresponding Map and Reduce functions for the following query:

Compute the average invoice total of all invoices from Germany.

Note: The input to the map function is *one* document collection which contains all objects of the database, regardless of their type. *Hint:* it is also possible, to create multiple map / reduce functions, where the output of the previous reduce task is the input of the next map task.Provide a sketch for the MapReduce implementation of these queries using some form of Java/JavaScript pseudo code, similar to the syntax used in the lecture.