Assignment 2: Benchmarking

NSM vs. DSM and Isolation Levels

Special Topics in Computer Science: Big Data Systems
CS-UH 3260 Spring 2023
MAX 10 points

This assignment contains two parts. The first focuses on analyzing the performance of two popular DBMSs with different storage models (DSM vs NSM). The second part involves simulating concurrency control issues.

PART I: In this assignment, you will evaluate two popular DBMSs featuring different storage models (DSM vs NSM), in addition to other optimization technologies. The performance evaluation will be based on real-world data obtained from the US Bureau of Transportation Statistics http://www.transtats.bts.gov/, which provides information on domestic flights within the United States.

1. System installation (not graded):

- Install MySQL, if not already installed, and create a database called "flight"
- Install MonetDB and follow instructions in lab3_monet.readme
    - note: call your database called "flight"

2. Database initialization and loading:

Next, you are going to load data from BTS collected for the years 2021 and 2022 (~7GB). The data has been preprocessed and is ready for you to create and populate a table called "ontime" with over 100 columns.

➔ The data and the helper scripts are available on Drive.

A. Schema creation
   a. `time mysql -uroot -p -D flight --local-infile=1 < air_ddl.sql`
   b. `time mclient -u monetdb -d flight < air_ddl.sql`

B. Data Loading
   a. `time mysql -uroot -p -D flight --local-infile=1 < load_mysql.sql`
   b. `time mclient -u monetdb -d flight < load_monetdb.sql`

(Note: put the full path to the data in the load sql script.)

🎯**Deliverables:**

- Report loading times. **(0.5 point)**
- Get the table status in both system: **(0.5 point)**
  - MySQL
    - SHOW TABLE STATUS;
  - MonetDB
    - select * from ontime limit 1;
    - select count(*) as c from ontime;

3. Benchmarking:

A. Compare the execution time of MySQL and MonetDB to perform the SQL Queries[1] listed on the following page.

- ***Advise:*** *By default, the MySQL client provides the execution time of queries. However, in MonetDB, you need to activate the timer when starting the client using the following command:*
  - `mclient -u monetdb -d flight --timer=performance`

🎯**Deliverables:**

- Create histograms comparing the execution times of all five queries in MonetDB and MySQL, and analyze the performance of the systems based on their respective data storage models. **(4 points)**
- Propose ways to improve the performance of the slower system on one of the queries. **(1point)**

B. **Automated Benchmarking (optional).** As a stretch goal**,** create a script to automate the execution of all queries, the randomization of the variables, and the collection of the execution times.

🎯**Deliverable:**

- Scripts in your favorite language **(1 point 🙌)**

[1] The Queries are a subset of the Percona's Benchmark

Q1: Count records

```
SELECT count(*) form ontime;
```

Q2: Average monthly flights

```
SELECT avg(c1) FROM (
    SELECT YearD, MonthD, count(*) AS c1
    FROM ontime
    GROUP BY YearD, MonthD
) as tmp;
```

Q3: The number of flights per day from January to June in 2021

```
SELECT DayOfWeek, count(*) AS c
FROM ontime
WHERE YearD=2021 AND MonthD BETWEEN 1 AND 6
GROUP BY DayOfWeek
ORDER BY c DESC;
```

Q4: The number of delays by carrier for 2022

```
SELECT Carrier, count(*)
FROM ontime
WHERE DepDelay>10 AND YearD=2022
GROUP BY Carrier
ORDER BY count(*) DESC;
```

Q5: Most popular destination by count of direct connected flights

```
SELECT
  DestCityName AS destination_city,
  count(DISTINCT OriginCityName) AS num_origins_of_flights
FROM ontime
WHERE YearD BETWEEN 2021 AND 2021
GROUP BY DestCityName
ORDER BY 2 DESC
LIMIT 10;
```

PART II: In this exercise, you will simulate issues related to concurrency control by modifying the default isolation level of MySQL

In your MySQL instance, create a new database and a simple relation. You may use the following code, or any other schema of your choice:

- CREATE DATABASE transaction_iso;
- USE transaction_iso;
- CREATE TABLE t (a INT NOT NULL, b INT, c INT);
- INSERT INTO t VALUES (1,2,3),(2, 4, 5),(3,12,10);

Then, using two mysql clients, simulate the following scenarios.

- *Phantom read.*
- *Unrepeatable read.*
- *Dirty read.*
- *Lost update.*

🎯**Deliverable:**

- Document the transactions that you performed to simulate each concurrency problem **(1 point each (total 4pts))**

Hints:

- Note your current isolation level using:
  - SHOW VARIABLES WHERE Variable_name LIKE "%_isolation";
- Make sure you are setting the sessions to the same isolation level. For example, to set the session into read uncommitted mode use:
  - SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
- Transactions are created, committed, and aborted as follows:
  - START TRANSACTION;
  - ….
  - COMMIT;
  - ROLLBACK;
- These are the isolation levels in mysql:
  - READ UNCOMMITTED
  - READ COMMITTED
  - REPEATABLE READ
  - SERIALIZABLE