HANDLING QUERIES ON STREAMING DATA

Paper in Reference:

Scalable Analytics on Fast Data

ANDREAS KIPF, VARUN PANDEY, and JAN BÖTTCHER, Technical University of Munich, Germany LUCAS BRAUN, Oracle Labs, Switzerland THOMAS NEUMANN and ALFONS KEMPER, Technical University of Munich, Germany

Faculty: Prof. Minal Bhise
TA: Ami Pandat
Mentor: Mayank Patel

MOTIVATION

The motivation behind this research was to make MMDB’s efficient enough so that they can be picked over hand crafted systems which are very costly to maintain. If MMDBs would offer better support for streaming workloads, it would make them well-suited for analytical streaming workloads.

The motivation behind this implementation is to answer user’s query in least possible time with the real time updates of the data using a main memory database system thereby handling the streaming data workload efficiently.

OBJECTIVE

The objective of the research paper was to explore how off-the-shelf Main memory database systems can be extended to sufficiently satisfy the requirements of analytics on fast data. Another intent of this paper is to close the gap between the extended versions of MMDB's and the modern streaming systems.

The objective of our implementation is to create and manage the database in such a way that we are able to answer the user's analytical queries on the basis of the combination of the historical data and the real time streaming data as fast as possible.

PROPOSED CONTRIBUTION

Streaming data is data that is generated continuously by thousands of data sources, which typically sent to the data records simultaneously in small sizes.

This data needs to be processed sequentially and is used for a wide variety of analytics.Data streams enable companies to use this real-time analytics to monitor their activities.

Handling such streaming data workloads and querying on them efficiently and optimally (minimal query response time) would offer numerous advantages that are transforming the way businesses and industries run.

LITERATURE SURVEY

Recent work on querying data streams has focused on systems where newly arriving data is processed and continuously streamed to the user in real time.

However, in many emerging applications, ad hoc queries also require the processing of data that arrives prior to query submission or during a period of disconnection. For such applications, PSoup, a system that combines the processing of ad hoc and continuous queries by treating data and queries symmetrically, allowing new queries to be applied to old data and new data to be applied to old queries is developed.

Handling streaming queries over streaming data is yet another related domain which describes the design and implementation of a novel query engine (PSoup) that treats data and query streams analogously, and performs multi query evaluation by joining them.

PROPOSED TECHNIQUE

- Data streaming is the process of transmitting a continuous flow of data. This data keeps on increasing with time as more and more real time data keeps getting updated.

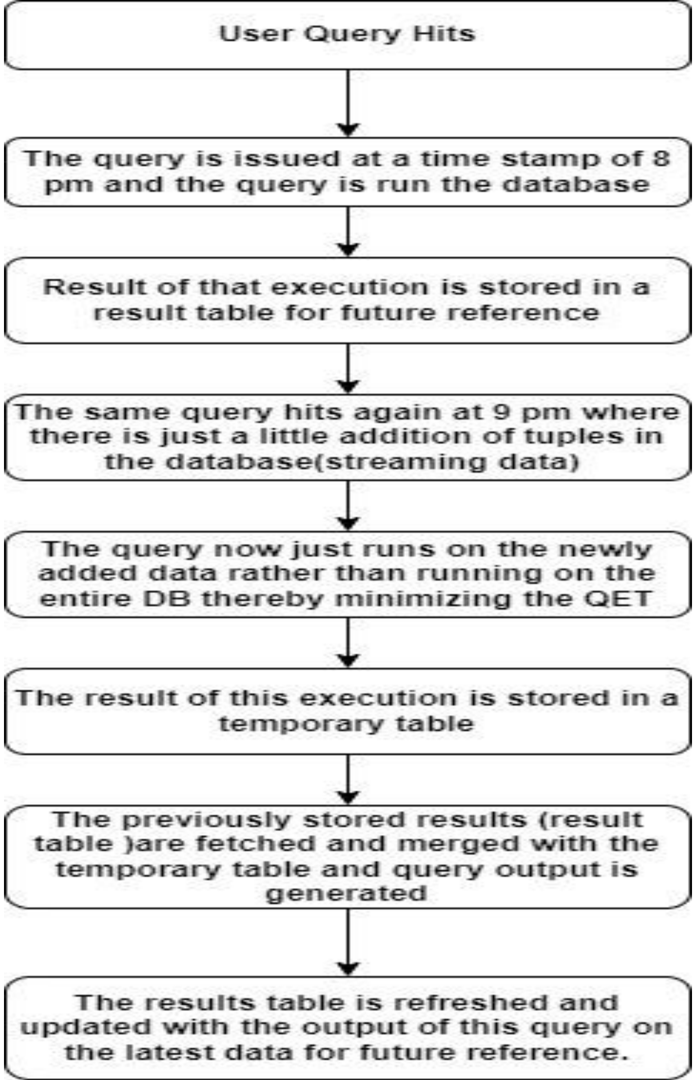
- The user queries have to run on very large datasets as the data keeps coming in real time, increasing the size of the database every minute thereby making the execution of queries time consuming.

- In our implementation, we hit the queries in such a way that the overall response time of the queries is optimised and the results are generated as quickly as possible.

- A certain query runs on a database and gives the result to the user. During this process it takes a certain amount of time that we call **QET.**

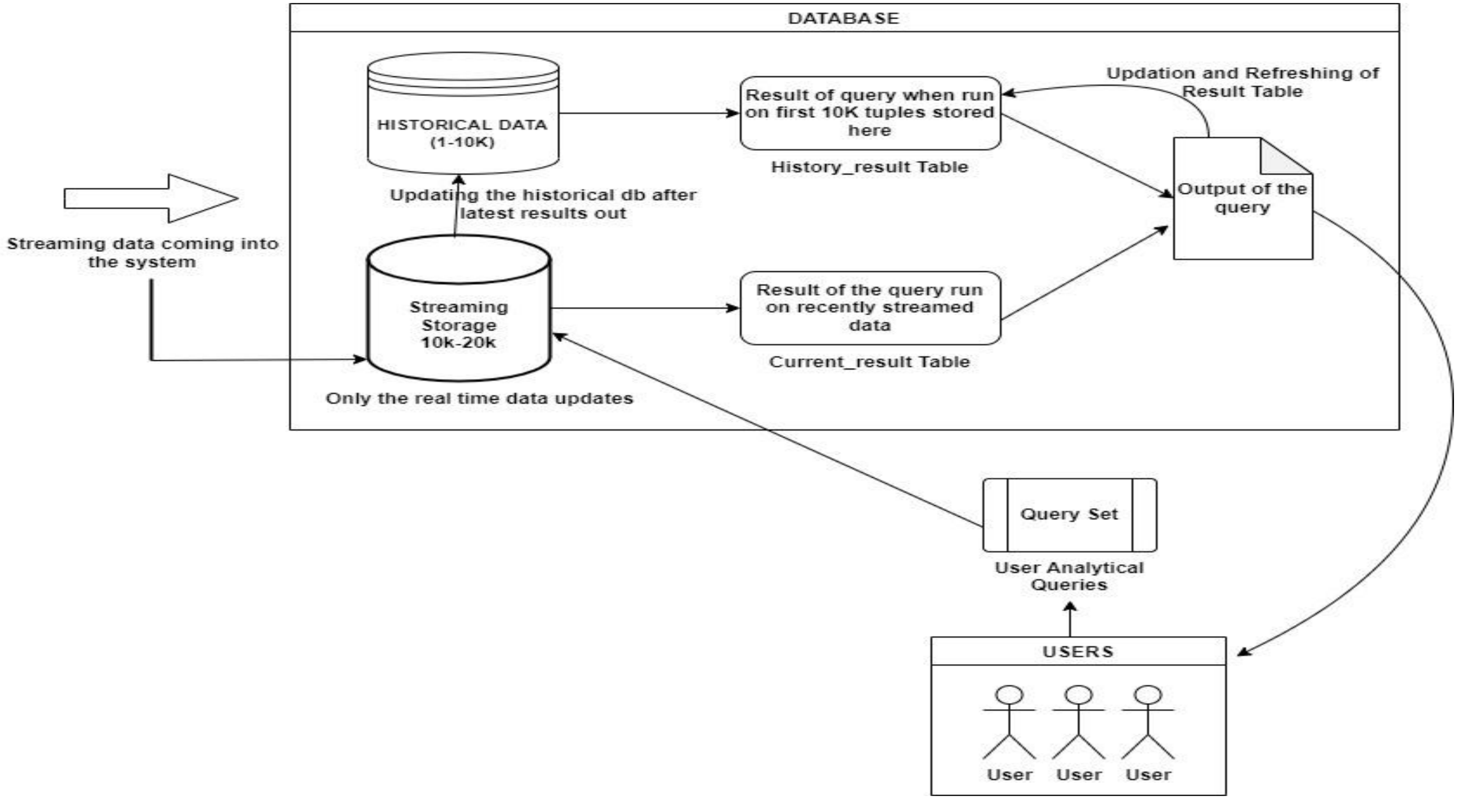
- Since we are talking about the streaming data and dealing with the real time updates on that, there will be an addition of data to the existing database after every few minutes/seconds.

- When the same query hits again, in the conventional method, we used to run the query again, on the updated database which now has a larger size than before leading to a greater QET.



- In our approach, when we run a query on a certain amount of data, we store those results and the next time the same query is put up again, we just run the query on the new addition of data (streaming data) and combine its result with the past stored results of the historical data and display it to the user.

WORKFLOW DIAGRAM



PROPOSED EXPERIMENTAL SETUP

- Dataset - Parking Data Stream, Aarhus, Denmark

  http://iot.ee.surrey.ac.uk:8080/datasets.html

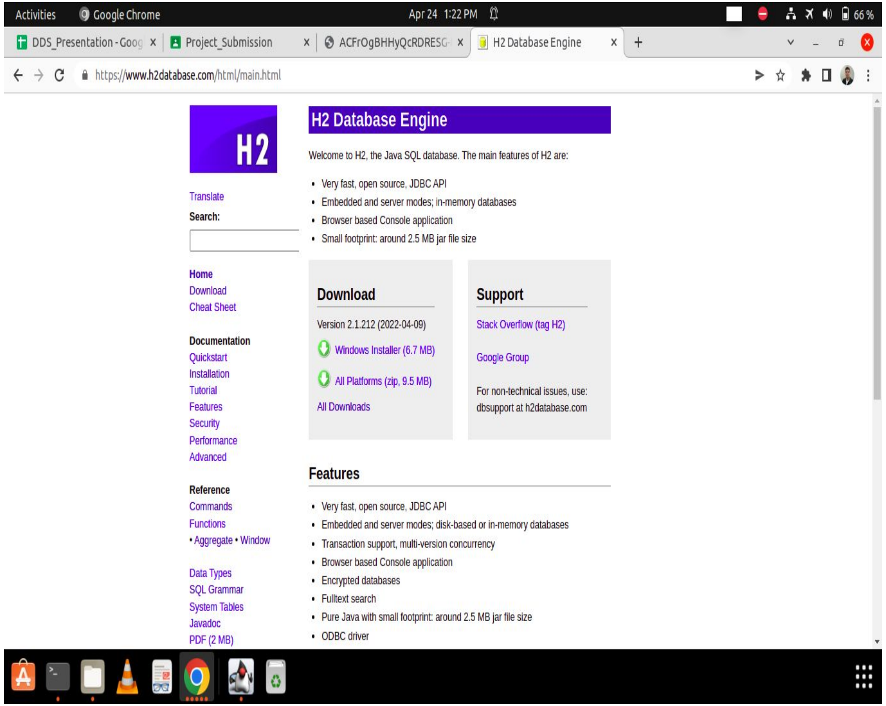
- Software - H2 is an open-source lightweight Java database which primarily deals with relational data..

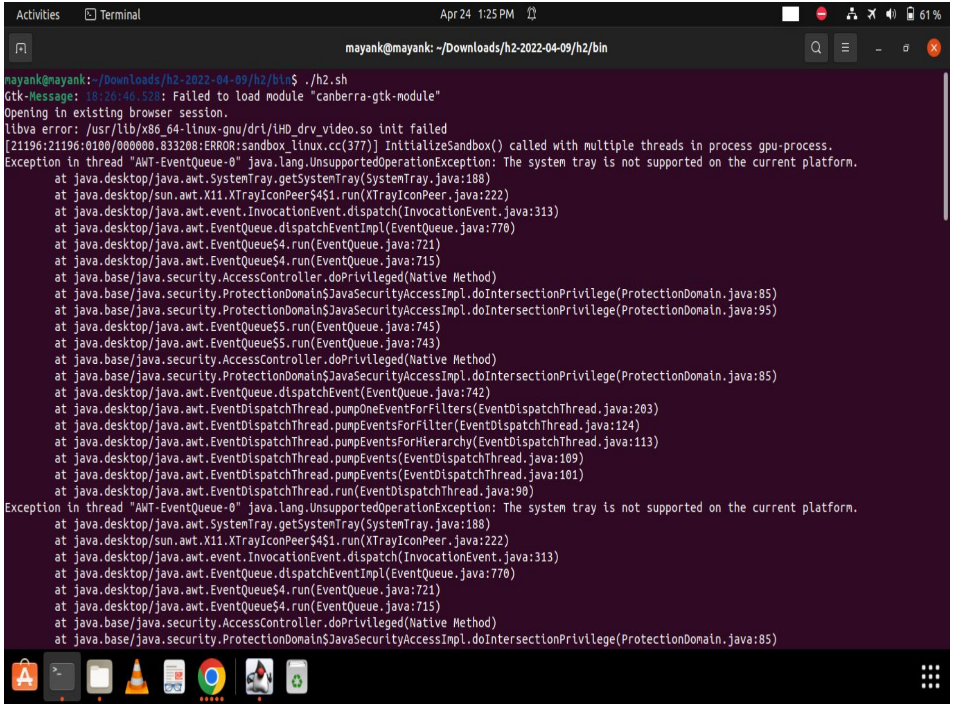
- Hardware - Linux Ubuntu 21.10, Intel Core i3, CPU@1.70GHz x 4, OS Type = 64 bit, RAM = 4GB.

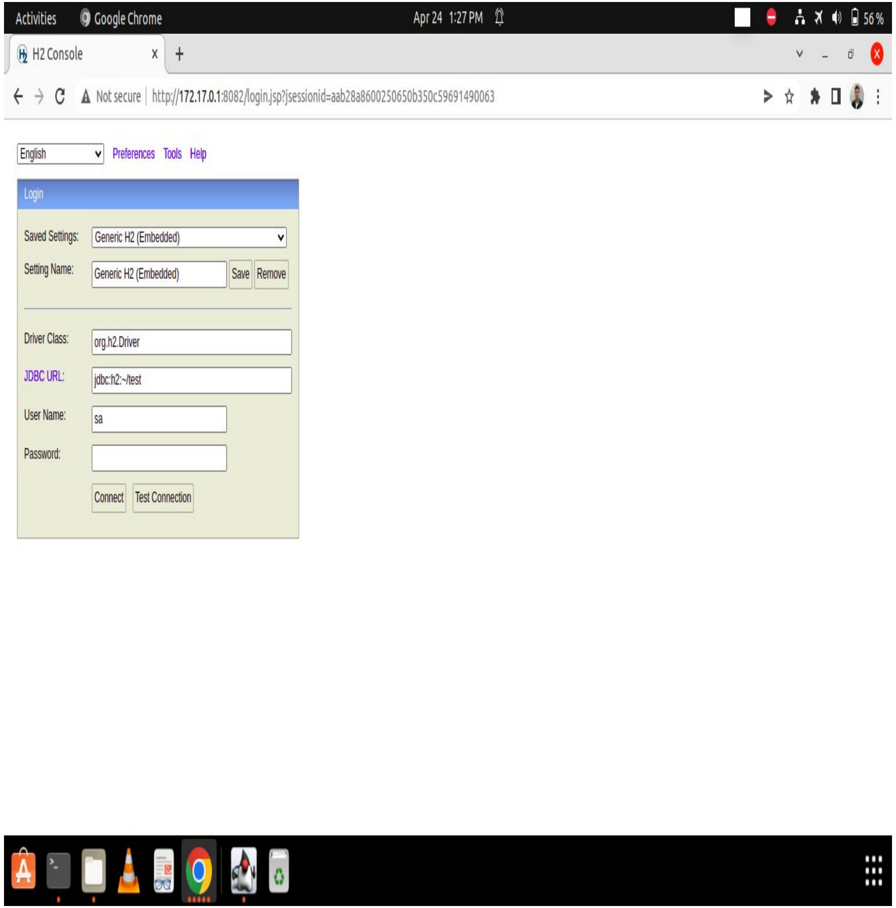
QUERY SET

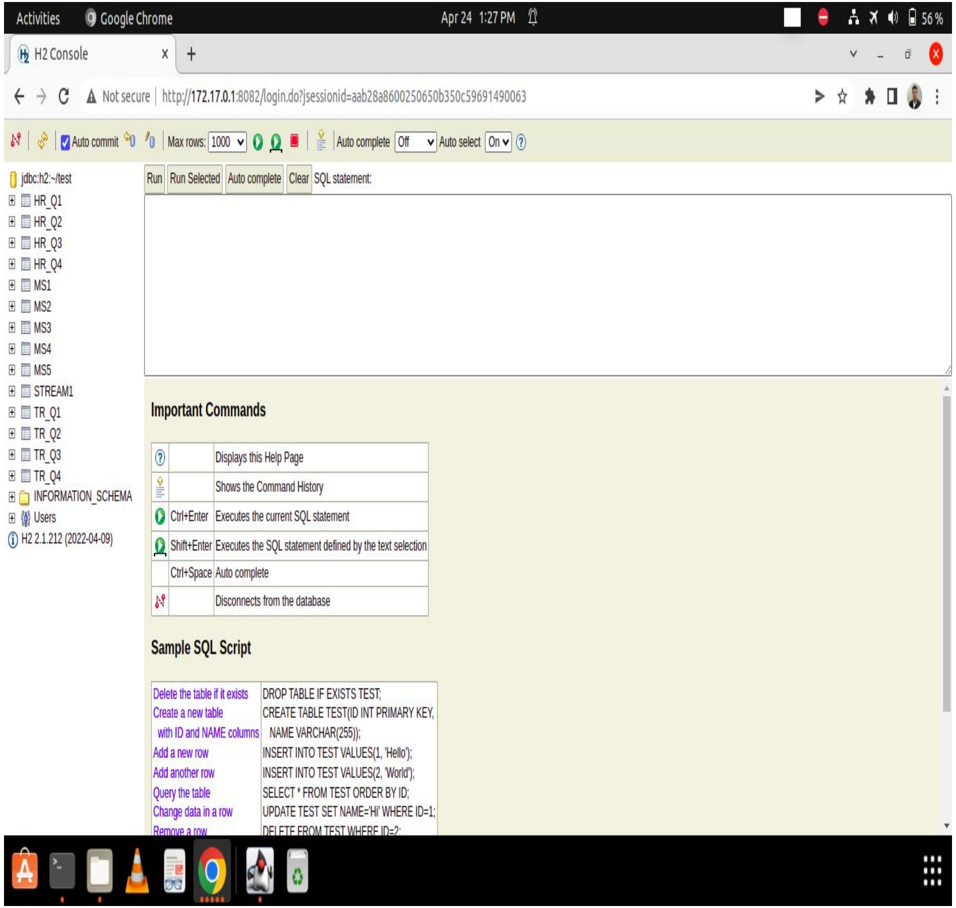
- **Query 1: Display distinct garages available for parking.**
  - SELECT DISTINCT (garagecode) FROM stream1
- **QUERY 2: Display the name of the garages that are presently vacant.**
  - SELECT DISTINCT garagecode FROM stream1 WHERE vehiclecount=0
- **QUERY 3: Display the name of the garages that are extremely crowded with vehicle count>=700.**
  - SELECT DISTINCT garagecode FROM stream1 WHERE vehiclecount>=700
- **QUERY 4: Display the name of garages that are overloaded and the parking is full.**
  - SELECT DISTINCT garagecode FROM stream1 WHERE vehiclecount>=totalspaces

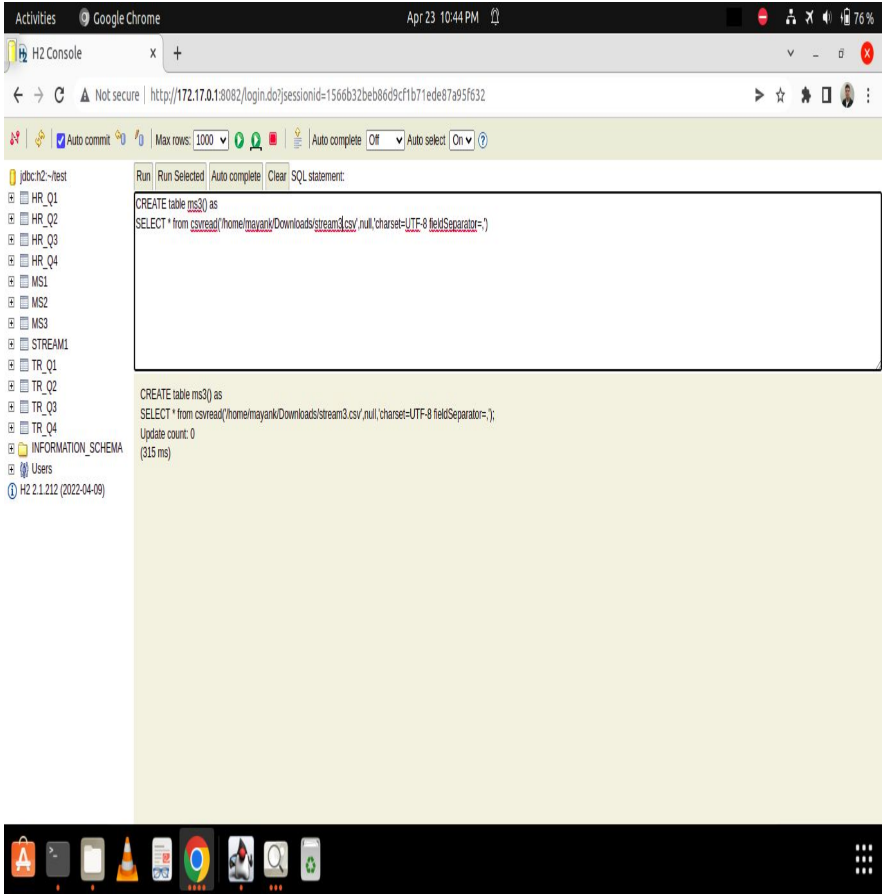
EXPERIMENTAL SETUP

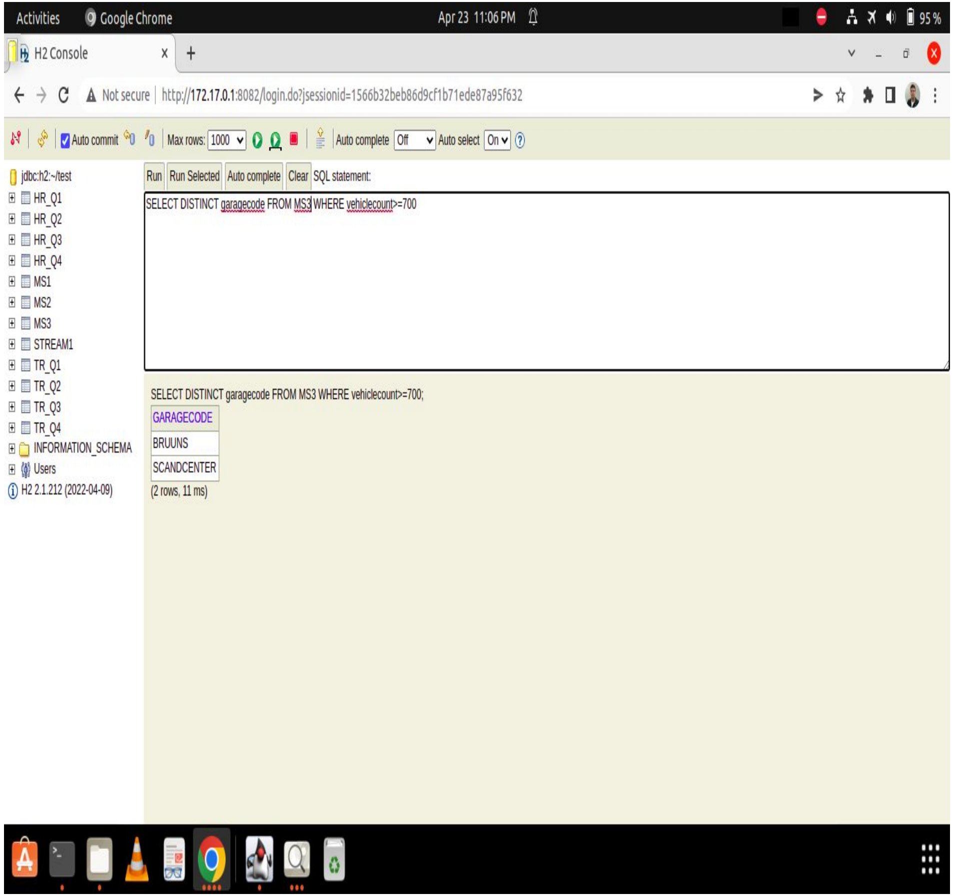




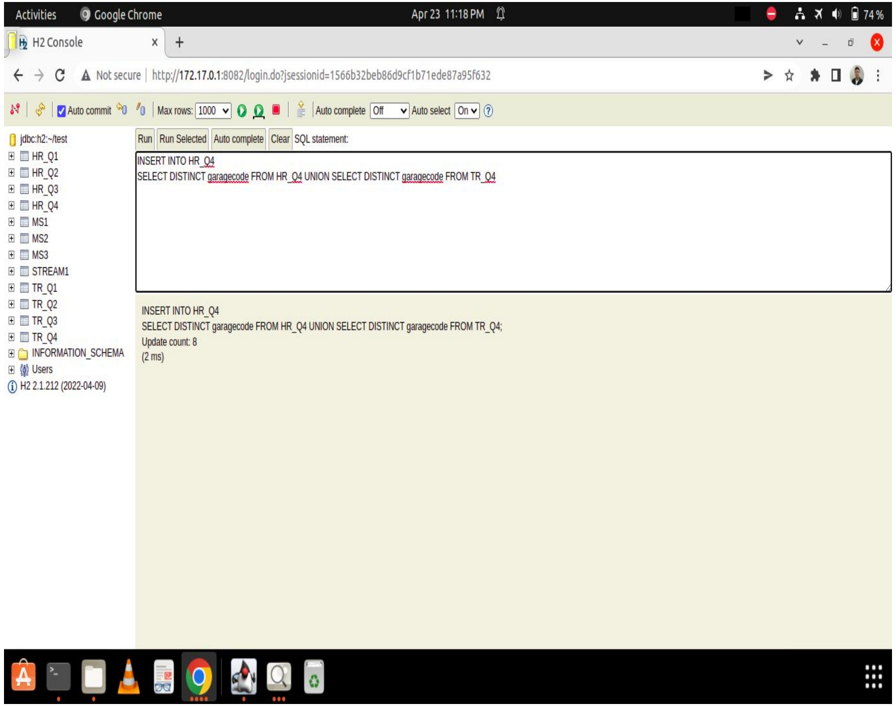

Activities
Google Chrome
Apr 24 1:27 PM
56 %
H2 Console
Not secure
http://172.17.0.1:8082/login.jsp?jsessionid=aab28a8600250650b350c59691490063
English
Preferences
Tools
Help
Login
Saved Settings:
Generic H2 (Embedded)
Setting Name:
Generic H2 (Embedded)
Save
Remove
Driver Class:
org.h2.Driver
JDBC URL:
jdbc:h2:~/test
User Name:
sa
Password:
Connect
Test Connection

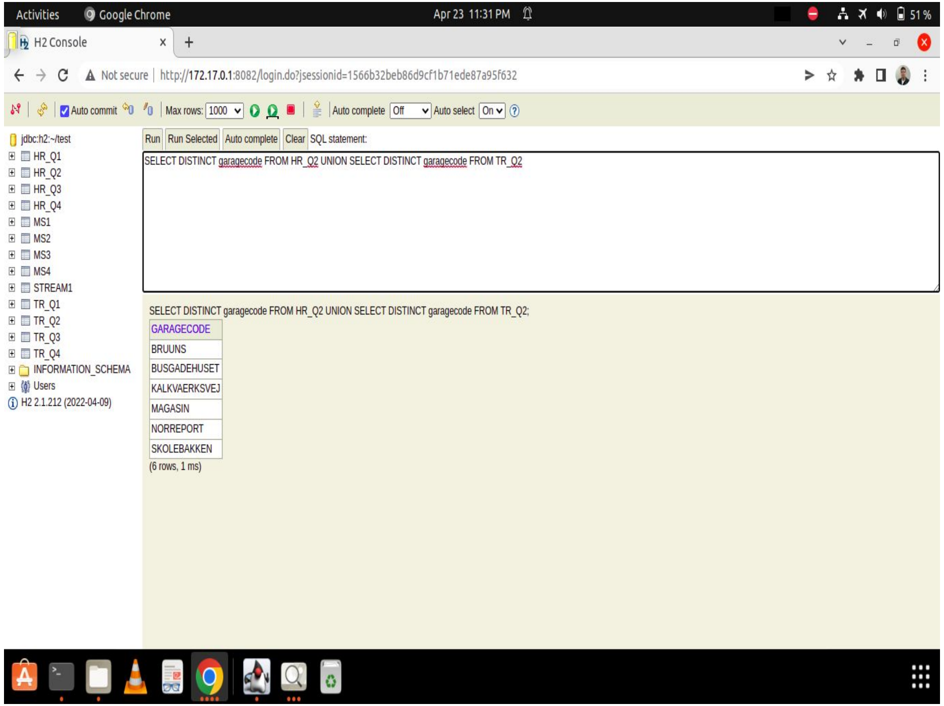

Activities
Google Chrome
Apr 24 1:27 PM
56 %
H2 Console
Not secure
http://172.17.0.1:8082/login.do?jsessionid=aab28a8600250650b350c59691490063
Auto commit
Max rows:
1000
Auto complete
Off
Auto select
On
Run
Run Selected
Auto complete
Clear
SQL statement:
jdbc:h2:~/test
HR_Q1
HR_Q2
HR_Q3
HR_Q4
MS1
MS2
MS3
MS4
MS5
STREAM1
TR_Q1
TR_Q2
TR_Q3
TR_Q4
INFORMATION_SCHEMA
Users
H2 2.1.212 (2022-04-09)
Important Commands
Displays this Help Page
Shows the Command History
Ctrl+Enter
Executes the current SQL statement
Shift+Enter
Executes the SQL statement defined by the text selection
Ctrl+Space
Auto complete
Disconnects from the database
Sample SQL Script
Delete the table if it exists
DROP TABLE IF EXISTS TEST;
Create a new table
with ID and NAME columns
CREATE TABLE TEST(ID INT PRIMARY KEY,
NAME VARCHAR(255));
Add a new row
INSERT INTO TEST VALUES(1, 'Hello');
Add another row
INSERT INTO TEST VALUES(2, 'World');
Query the table
SELECT * FROM TEST ORDER BY ID;
Change data in a row
UPDATE TEST SET NAME='Hi' WHERE ID=1;


Activities
Google Chrome
Apr 23 10:44 PM
76 %
H2 Console
Not secure
http://172.17.0.1:8082/login.do?jsessionid=1566b32beb86d9cf1b71ede87a95f632
Auto commit
Max rows: 1000
Auto complete Off
Auto select On
jdbc:h2:~/test
HR_Q1
HR_Q2
HR_Q3
HR_Q4
MS1
MS2
MS3
STREAM1
TR_Q1
TR_Q2
TR_Q3
TR_Q4
INFORMATION_SCHEMA
Users
H2 2.1.212 (2022-04-09)
Run
Run Selected
Auto complete
Clear
SQL statement:
CREATE table ms3() as
SELECT * from csvread('/home/mayank/Downloads/stream3.csv',null,'charset=UTF-8 fieldSeparator=,')
CREATE table ms3() as
SELECT * from csvread('/home/mayank/Downloads/stream3.csv',null,'charset=UTF-8 fieldSeparator=,');
Update count: 0
(315 ms)


Activities
Google Chrome
Apr 23 11:06 PM
95 %
H2 Console
Not secure
http://172.17.0.1:8082/login.do?jsessionid=1566b32beb86d9cf1b71ede87a95f632
Auto commit
Max rows: 1000
Auto complete Off
Auto select On
jdbc:h2:~/test
HR_Q1
HR_Q2
HR_Q3
HR_Q4
MS1
MS2
MS3
STREAM1
TR_Q1
TR_Q2
TR_Q3
TR_Q4
INFORMATION_SCHEMA
Users
H2 2.1.212 (2022-04-09)
Run
Run Selected
Auto complete
Clear
SQL statement:
SELECT DISTINCT garagecode FROM MS3 WHERE vehiclecount>=700
SELECT DISTINCT garagecode FROM MS3 WHERE vehiclecount>=700;
GARAGECODE
BRUUNS
SCANDCENTER
(2 rows, 11 ms)

EXPERIMENTAL RUNS





EVALUATION & RESULTS

- The system with the following specifications were used for the implementation
  - Linux Ubuntu 21.10, Intel Core i3, CPU@1.70GHz x 4, OS Type = 64 bit, RAM = 4GB.
- A set of 4 queries is run on the Parking dataset and the evaluation of our technique is on the basis of the following parameters.
    - Query Execution Time (QET)
    - Data Storing Time (DST)
    - Query Response Time (QRT)

RESULTS & CONCLUSION

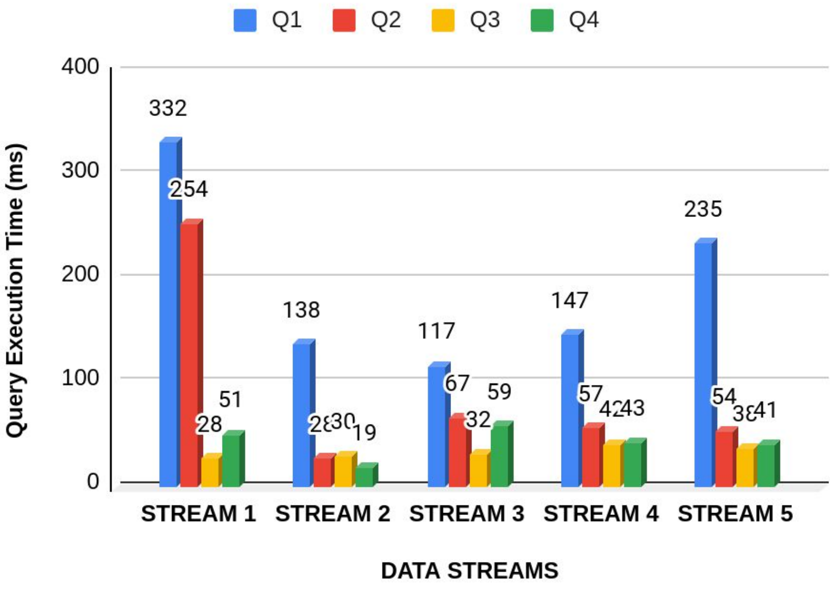
- The Query Execution Time in our approach is considerably lower than that of the Conventional Method which was the actual bottleneck of the problem.

- We were able to successfully reduce the overall Query Response Time thereby showing an upperhand of our method over the conventional one.

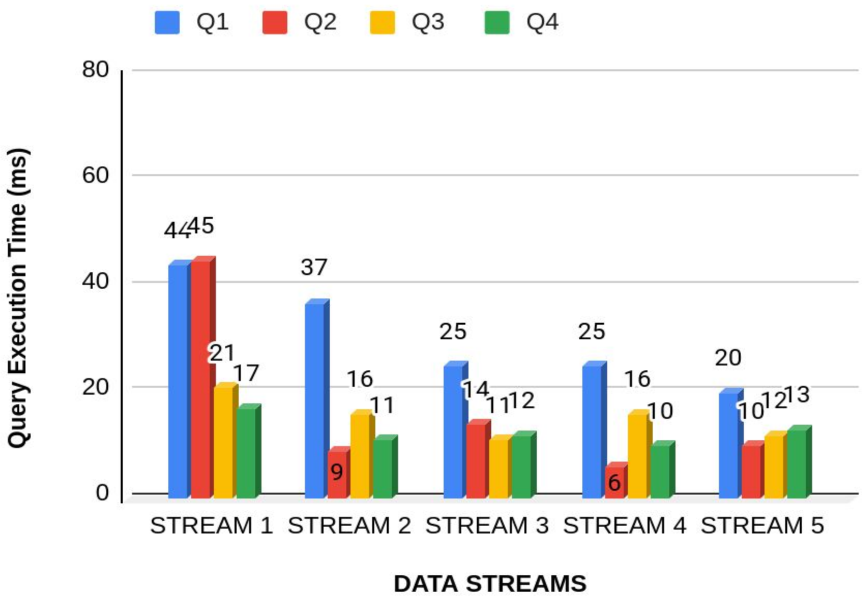
- Conventional: QRT= QET

  Our Approach: QRT =QET** + DST + FMT(fetching & merging time)

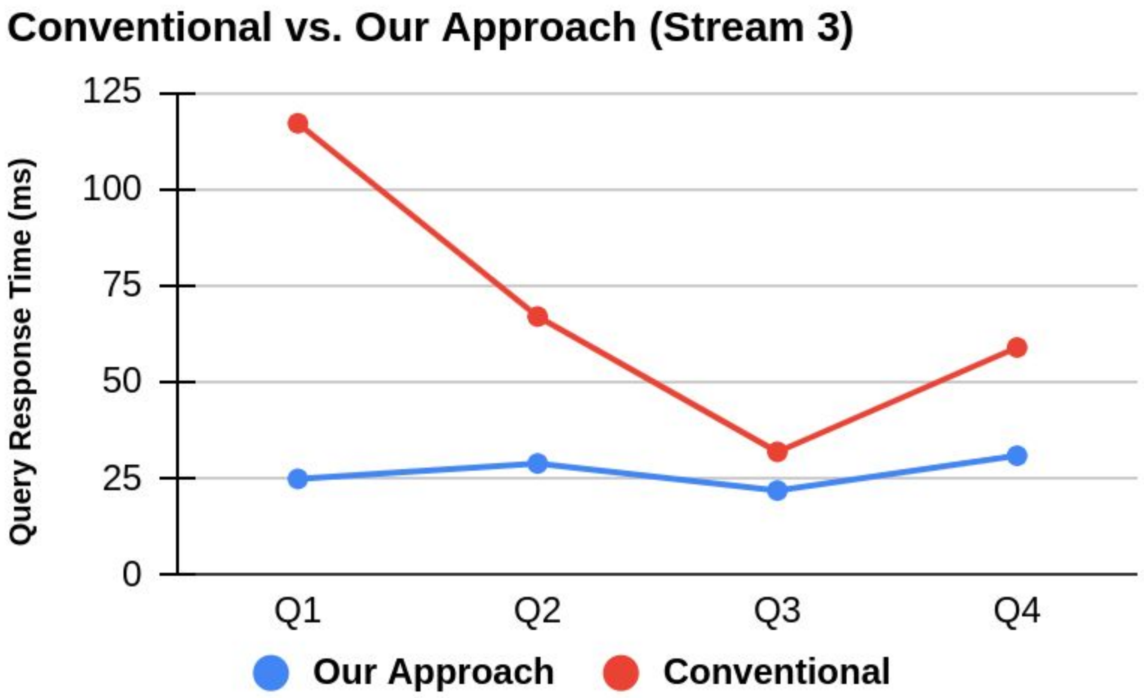
Conventional Approach



Our Implemenation



A substantial decrease in the QET because of the reduced dataset on which the query is running.



On comparing the <u>overall query response time</u> for the data stream 3, our technique gave better results..

CONCLUSION

- A substantial decrease in the QET of queries is witnessed in our technique as our method reduces the dataset on which the query is running.

- Our approach gives better results of the query response time on datastreams when compared with the conventional approach

REFERENCES

- https://www.h2database.com/html/main.html
- https://www.h2database.com/html/installation.html
- https://www.h2database.com/html/tutorial.html
- https://www.h2database.com/html/commands.html
- https://www.researchgate.net/publication/329736601_Query_Processing_for_Streaming_RDF_Data
- https://www.tibco.com/reference-center/what-is-data-streaming
- https://www.infoworld.com/article/3646589/what-is-streaming-data-event-stream-processing-explained.html
- https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=queries+on+streaming+data&btnG=

THANK YOU

MAYANK SINGH-202111019
PRIYANKA MISHRA-202111056