



Experiment No. 3

Title: Execution of In-memory database queries



Batch: A2 Roll No.: 16010421059

Experiment No.:2

Aim: To execute In-memory database queries**Resources needed: MySQL**

Theory

In-Memory database is a database that uses a system's main memory for data storage rather than the disk-based storage typically utilized by traditional databases. In-memory databases, or IMDBs, are frequently employed in high-volume environments where response time is critical, as access times and database requests are typically considerably faster when system memory is used as opposed to hard disk storage.

The traditional databases and in-memory databases can be used together and referred as hybrid databases, which support both in-memory and disk-based storage in order to maximize performance as well as reliability of the system. All most all RDBMS systems available in market supports In-Memory databases.

MySQL In-Memory database:

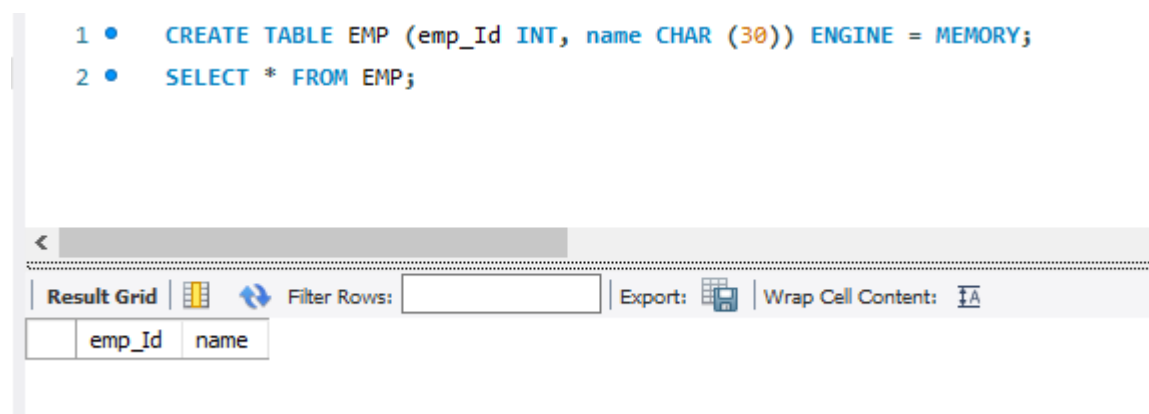
In MySQL DB, the MEMORY storage engine creates special-purpose tables with contents that are stored in memory. Because the data is vulnerable to crashes, hardware issues, or power outages, use of these tables are limited to temporary work areas or read-only caches for data pulled from other tables.

A typical use case for the MEMORY engine involves these characteristics:

- Operations involving transient, non-critical data such as session management or caching. When the MySQL server halts or restarts, the data in MEMORY tables is lost.
- In-memory storage for fast access and low latency. Data volume can fit entirely in memory without causing the operating system to swap out virtual memory pages.
- A read-only or read-mostly data access pattern (limited updates).
- MEMORY tables cannot contain BLOB or TEXT columns.





To create a MEMORY table, specify the clause ENGINE=MEMORY on the CREATE TABLE statement

```
CREATE TABLE EMP (emp_Id INT, name CHAR (30)) ENGINE = MEMORY;
```



As indicated by the engine name, MEMORY tables are stored in memory. They use hash indexes by default, which makes them very fast for single-value lookups, and very useful for creating temporary tables. However, when the server shuts down, all rows stored in MEMORY tables are lost. The tables themselves continue to exist because their definitions are stored in .frm files on disk, but they are empty when the server restarts.

```
1 • CREATE TABLE EMP (emp_Id INT, name CHAR (30)) ENGINE = MEMORY;
2 • SELECT * FROM EMP;
3 • INSERT INTO EMP
4   VALUES('10','ABC');
5 • INSERT INTO EMP
```

<   Filter Rows: Export:  Wrap Cell Content: 

	emp_Id	name
▶	10	ABC
	20	DEF
	30	GHI
	40	JKL

To load the data in memory from other existing table use,

```
CREATE TABLE EMP (emp_Id INT, name CHAR (30)) ENGINE=MEMORY
as SELECT * FROM EMP;
```

To move the data from In-Memory table to hard drive (using any text file) use the following syntax,

```
SELECT * INTO OUTFILE 'emp_data.txt' FROM EMP;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the query: `SELECT * INTO OUTFILE 'emp_data.txt' FROM EMP;`. The output pane displays a table with columns: #, Time, Action, Message, and Duration / Fetch. The table contains 9 rows of execution logs, including the successful execution of the SELECT statement into the outfile.

#	Time	Action	Message	Duration / Fetch
1	13:31:15	Apply changes to 059_Chinnmay	Changes applied	
2	13:35:35	CREATE TABLE EMP (emp_Id INT, name CHAR (30)) ENGINE = MEMORY	0 row(s) affected	0.031 sec
3	13:37:10	SELECT * FROM EMP LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
4	13:44:10	INSERT INTO EMP VALUES(10,'ABC')	1 row(s) affected	0.047 sec
5	13:44:10	INSERT INTO EMP VALUES(20,'DEF')	1 row(s) affected	0.000 sec
6	13:44:10	INSERT INTO EMP VALUES(30,'GHI')	1 row(s) affected	0.000 sec
7	13:44:10	INSERT INTO EMP VALUES(40,'JKL')	1 row(s) affected	0.000 sec
8	13:45:06	SELECT * FROM EMP LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
9	13:46:52	SELECT * INTO OUTFILE 'emp_data.txt' FROM EMP	4 row(s) affected	0.016 sec

To populate a MEMORY table when the MySQL server starts, use the INFILE option. For example,

```
LOAD DATA INFILE 'emp_data.txt' INTO TABLE EMP;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the query: `LOAD DATA INFILE 'emp_data.txt' INTO TABLE EMP;`. The output pane displays a table with columns: #, Time, Action, Message, and Duration / Fetch. The table contains 10 rows of execution logs, including the successful execution of the LOAD DATA INFILE statement.

#	Time	Action	Message	Duration / Fetch
1	13:31:15	Apply changes to 059_Chinnmay	Changes applied	
2	13:35:35	CREATE TABLE EMP (emp_Id INT, name CHAR (30)) ENGINE = MEMORY	0 row(s) affected	0.031 sec
3	13:37:10	SELECT * FROM EMP LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
4	13:44:10	INSERT INTO EMP VALUES(10,'ABC')	1 row(s) affected	0.047 sec
5	13:44:10	INSERT INTO EMP VALUES(20,'DEF')	1 row(s) affected	0.000 sec
6	13:44:10	INSERT INTO EMP VALUES(30,'GHI')	1 row(s) affected	0.000 sec
7	13:44:10	INSERT INTO EMP VALUES(40,'JKL')	1 row(s) affected	0.000 sec
8	13:45:06	SELECT * FROM EMP LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
9	13:46:52	SELECT * INTO OUTFILE 'emp_data.txt' FROM EMP	4 row(s) affected	0.016 sec
10	13:48:23	LOAD DATA INFILE 'emp_data.txt' INTO TABLE EMP	4 row(s) affected Records: 4 Deleted: 0 Skipped: 0 Warnings: 0	0.015 sec

Where, emp_data.txt is a data file.

Procedure:

Perform following tasks:

1. Create In-memory table using Engine as Memory.
2. Insert values in that table.
3. Attempt to retrieve values from the table after restarting the database server.
4. Load the data into table using file load.

Results: (Program printout with output)

```
1 • CREATE TABLE EMP (emp_Id INT, name CHAR (30)) ENGINE = MEMORY;
2 • SELECT * FROM EMP;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the query: `SELECT * FROM EMP;`. The result grid displays the output of the query, showing two columns: emp_Id and name.

emp_Id	name
10	ABC
20	DEF
30	GHI
40	JKL

```

1 • CREATE TABLE EMP (emp_Id INT, name CHAR (30)) ENGINE = MEMORY;
2 • SELECT * FROM EMP;
3 • INSERT INTO EMP
4   VALUES('10','ABC');
5 • INSERT INTO EMP

```

Result Grid

emp_Id	name
10	ABC
20	DEF
30	GHI
40	JKL

```

14 • LOAD DATA INFILE 'emp_data.txt' INTO TABLE EMP;

```

Output

#	Time	Action	Message	Duration / Fetch
1	13:31:15	Apply changes to 059_Chinmay	Changes applied	
2	13:35:35	CREATE TABLE EMP (emp_Id INT, name CHAR (30)) ENGINE = MEMORY	0 row(s) affected	0.031 sec
3	13:37:10	SELECT * FROM EMP LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
4	13:44:10	INSERT INTO EMP VALUES('10','ABC')	1 row(s) affected	0.047 sec
5	13:44:10	INSERT INTO EMP VALUES('20','DEF')	1 row(s) affected	0.000 sec
6	13:44:10	INSERT INTO EMP VALUES('30','GHI')	1 row(s) affected	0.000 sec
7	13:44:10	INSERT INTO EMP VALUES('40','JKL')	1 row(s) affected	0.000 sec
8	13:45:06	SELECT * FROM EMP LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
9	13:46:52	SELECT * INTO OUTFILE 'emp_data.txt' FROM EMP	4 row(s) affected	0.016 sec
10	13:48:23	LOAD DATA INFILE 'emp_data.txt' INTO TABLE EMP	4 row(s) affected Records: 4 Deleted: 0 Skipped: 0 Warnings: 0	0.015 sec

```

1 • CREATE TABLE EMP (emp_Id INT, name CHAR (30)) ENGINE = MEMORY;
2 • SELECT * FROM EMP;
3 • INSERT INTO EMP
4   VALUES('10','ABC');
5 • INSERT INTO EMP

```

Result Grid

emp_Id	name
10	ABC
20	DEF
30	GHI
40	JKL
10	ABC
20	DEF
30	GHI

Questions:

1. What is the difference between traditional and In-memory databases?

Traditional Databases

- All data stored on disk, disk I/O needed to move data into main memory when needed.
- Data is always persisted to disk.
- Traditional data structures like B-Trees designed to store tables and indices efficiently on disk.
- Virtually unlimited database size.
- Support very broad set of workloads, i.e. OLTP, data warehousing, mixed workloads, etc.

In-Memory Databases

- All data stored in main memory, no need to perform disk I/O to query or update data.

- Data is persistent or volatile depending on the in-memory database product.
- Specialized data structures and index structures assume data is always in main memory.
- Optimized for specialized workloads; i.e. communications industry-specific HLR/HSS workloads.
- Database size limited by the amount of main memory.

2. List applications using in-memory database. Explain any one of it stressing upon advantage of using in-memory database.

Ans In-memory databases are ideal for applications that require microsecond response times or have large spikes in traffic such as gaming leaderboards, session stores, and real-time analytics.

Real-Time Analytics: Businesses need to implement tight operational feedback loops so decision makers can refine strategies quickly. In-memory databases support rapid iteration by removing conventional database bottlenecks like disk latency and CPU contention. Analysts appreciate the ability to get immediate data access with preferred analysis and visualization tools.

Outcomes:

CO1: Design advanced database systems using Parallel, Distributed and In-memory Databases and its implementation.

Conclusion: (Conclusion to be based on outcomes achieved)

From this experiment we have learnt about In-Memory Databases and successfully created database, created table, inserted values, loaded data from file and moved data into hard drive, using command prompt and XAMPP control panel we successfully setup SQL connection and performed all operations aforementioned.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:

1. <https://dev.mysql.com/doc/refman/5.5/en/memory-storage-engine.html>
2. <http://opensourceforu.ifytimes.com/2012/01/importance-of-in-memory-databases/>
3. <http://pages.cs.wisc.edu/~jhuang/qual/main-memory-db-overview.pdf>
4. <http://docs.memsql.com>