

VOLTDDB

VoltDB is an in-memory database management system. It enables processing data at very high speeds by keeping the data in memory (RAM).

VoltDB falls into a category known as NewSQL, offering a combination of relational database functionality with scalability and high performance.

The key features and advantages of VoltDB are as follows:

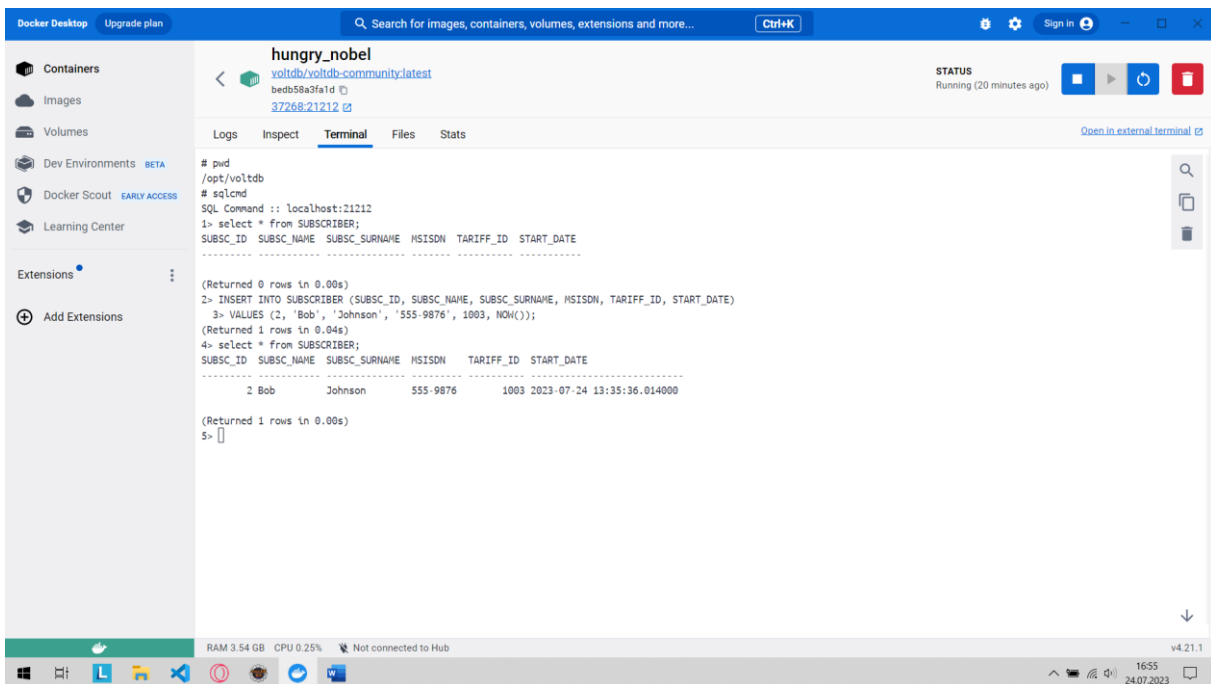
1. **In-Memory Processing:** VoltDB operates primarily in-memory, which allows for lightning-fast data access and processing.
2. **Horizontally Scalable:** It offers excellent horizontal scalability, enabling the database to handle increasing workloads by adding more nodes to the cluster.
3. **ACID Transactions:** VoltDB supports ACID (Atomicity, Consistency, Isolation, Durability) transactions, ensuring data integrity and reliability.
4. **Real-Time Analytics:** The in-memory architecture facilitates real-time analytics, making it suitable for applications requiring quick data analysis.
5. **High Throughput:** VoltDB's in-memory design, along with its optimized query execution, ensures high throughput and low-latency data processing.
6. **NoSQL and SQL Support:** It provides both NoSQL and SQL capabilities, allowing developers to choose the most suitable data model for their applications.
7. **Automatic Data Partitioning:** VoltDB automatically partitions data across nodes, optimizing data distribution and enhancing performance.
8. **Fault Tolerance:** The database system includes built-in fault tolerance mechanisms, ensuring data availability and preventing single points of failure.
9. **Stream Processing:** VoltDB supports stream processing, enabling real-time analysis of data streams and event-driven applications.

10. Simplified Development: Developers benefit from the familiar SQL interface, making it easier to develop applications without a steep learning curve.
11. Multi-Language Support: VoltDB supports various programming languages, providing flexibility for developers to work in their preferred environments.
12. Cloud-Ready: It is designed to work seamlessly in cloud environments, taking advantage of cloud resources and elasticity.
13. In-Memory Snapshot Backups: The ability to take in-memory snapshot backups ensures data resilience and quick recovery options.

Overall, VoltDB's features and advantages make it an attractive choice for high-performance, scalable, and real-time data-intensive applications.

VOLTDB CONNECTION WITH JAVA

VoltDB in Docker



```
Docker Desktop Upgrade plan Search for images, containers, volumes, extensions and more... Ctrl+K Sign in
```

hungry_nobel
voldb/voldb-community:latest
bedb5ba3fa1d
37268:21212

STATUS
Running (20 minutes ago)

Logs Inspect **Terminal** Files Stats Open in external terminal

```
# pwd  
/opt/voldb  
# sqlcmd  
SQL Command :: localhost:21212  
1> select * from SUBSCRIBER;  
SUBSC_ID SUBSC_NAME SUBSC_SURNAME HSISDN TARIFF_ID START_DATE  
-----  
(Returned 0 rows in 0.00s)  
2> INSERT INTO SUBSCRIBER (SUBSC_ID, SUBSC_NAME, SUBSC_SURNAME, HSISDN, TARIFF_ID, START_DATE)  
3> VALUES (2, 'Bob', 'Johnson', '555-9876', 1003, NOW());  
(Returned 1 rows in 0.04s)  
4> select * from SUBSCRIBER;  
SUBSC_ID SUBSC_NAME SUBSC_SURNAME HSISDN TARIFF_ID START_DATE  
-----  
2 Bob Johnson 555-9876 1003 2023-07-24 13:35:36.014000  
(Returned 1 rows in 0.00s)  
5>
```

RAM 3.54 GB CPU 0.25% Not connected to Hub v4.21.1
1655 24.07.2023

Java App

The screenshot shows the Eclipse IDE with a project named 'voltdb_learn'. The 'Project Explorer' on the left shows the project structure, including 'src/main/java' and 'src/test/java'. The 'Main.java' file is open in the editor, showing the following code:

```
1  import java.sql.SQLException;
2  import java.sql.Statement;
3
4  public class Main {
5
6      public static void main(String[] args) throws SQLException {
7
8          try {
9              // Load the VoltDB JDBC driver
10             Class.forName("org.voltdb.jdbc.Driver");
11             Connection connection = VoltDBConnection.getConnection();
12             System.out.println("Connected VoltDB");
13
14             Statement statement = connection.createStatement();
15             ResultSet resultSet = statement.executeQuery("SELECT * FROM SUBSCRIBER");
16             while (resultSet.next()) {
17                 int subscId = resultSet.getInt("SUBSC_ID");
18                 String subscName = resultSet.getString("SUBSC_NAME");
19                 System.out.println("SUBSC_ID: " + subscId + ", SUBSC_NAME: " + subscName);
20             }
21         } catch (ClassNotFoundException e) {
22             e.printStackTrace();
23         }
24     }
25 }
```

The 'Console' at the bottom shows the output of the application:

```
Summary: Main [D] [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (24 Tem 2023 16:54:25 - 16:54:26) [pid: 2672]
Connected VoltDB
SUBSC_ID: 2, SUBSC_NAME: Bob
```

The status bar at the bottom indicates the file is 'Writable', 'Smart Insert' is enabled, and the cursor is at line 4, column 154. The system clock shows 16:55 on 24.07.2023.