|  |  |
| --- | --- |
| **Topic** | HAZELCAST-ORACLE SQL COMPARISON |
| **Document Name** | HAZELCAST\_EX\_01 |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Difficulty Level** | | | |
| **Beginner** | **Junior** | **Senior** | **Expert** |
| □ | ■ | □ | □ |

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Ver | Comments |
| 26.08.2021 | Oktay Polat | 1.0 | Initial Draft |

# HAZELCAST-ORACLE SQL COMPARISON

## Definition

In this exercise you will compare execution time of Hazelcast and Oracle SQL.

**Hazelcast**

* Firstly, install Hazelcast IMDG.
* Put 20.000 random numbers with Hazelcast and take note the time.
* Then get those numbers randomly and take note the time.

\*Also do this for 100.000 random numbers.

**Oracle SQL**

* Firstly, insert 20.000 random numbers with Oracle SQL and take note the time.
* Then select those number randomly and take note the time.

\*Also do this for 100.000 random numbers.

At the end, you will have a table:

|  |  |  |
| --- | --- | --- |
|  | **Hazelcast** | **Oracle SQL** |
| **20.000 put/insert** | 694ms | 5563ms |
| **20.000 get/select** | 506ms | 2554ms |
| **100.000 put/insert** | 2065ms | 23423ms |
| **100.000 get/select** | 2065ms | 3095ms |

**Objectives** :

* To learn basics of Hazelcast.
* To see differences between Hazelcast data process and Oracle SQL data process.

**Exercise Keywords:** In Memory Data Grid, put, get, member, client, cluster.

## Solution of HAZELCAST-EX-01:

For solution of example, please send your answer to your supervisor.

OracleOps.java:

package org.example;  
  
import java.sql.\*;  
import java.util.Random;  
  
public class OracleOps {  
 private String dbUrl = "jdbc:oracle:thin:@//localhost:1521/TESTYVZ.kaan.com";  
 private String username = "C##KAAN";  
 private String password = "password";  
  
 public long insertRandomNumbers(int count) {  
 long startTime = System.*nanoTime*();  
 try (Connection connection = DriverManager.*getConnection*(dbUrl, username, password);  
 PreparedStatement preparedStatement = connection.prepareStatement("INSERT INTO RANDOM\_NUMBERS(NUMBER\_VALUE) VALUES(?)")) {  
  
 Random random = new Random();  
 for (int i = 0; i < count; i++) {  
 int randomNumber = random.nextInt(count);  
 preparedStatement.setString(1, String.*valueOf*(randomNumber));  
 preparedStatement.executeUpdate();  
 }  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }  
 long endTime = System.*nanoTime*();  
 return (endTime - startTime) / 1\_000\_000; // Convert to milliseconds  
 }  
  
 public long selectRandomNumbers() {  
 long startTime = System.*nanoTime*();  
 try (Connection connection = DriverManager.*getConnection*(dbUrl, username, password);  
 Statement statement = connection.createStatement()) {  
  
 String sql = "SELECT NUMBER\_VALUE FROM RANDOM\_NUMBERS";  
 try (ResultSet resultSet = statement.executeQuery(sql)) {  
 while (resultSet.next()) {  
 resultSet.getInt(1); // Fetch all values  
 }  
 }  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }  
 long endTime = System.*nanoTime*();  
 return (endTime - startTime) / 1\_000\_000; // Convert to milliseconds  
 }  
}

HazelcastOps.java:

package org.example;  
  
import com.hazelcast.core.Hazelcast;  
import com.hazelcast.core.HazelcastInstance;  
import com.hazelcast.map.IMap;  
  
import java.util.Random;  
  
public class HazelcastOps {  
 private final HazelcastInstance hazelcastInstance;  
 private final IMap<Integer, Integer> map;  
  
 public HazelcastOps() {  
 this.hazelcastInstance = Hazelcast.*newHazelcastInstance*();  
 this.map = hazelcastInstance.getMap("numbers");  
 }  
  
 public long putRandomNumbers(int count) {  
 long startTime = System.*nanoTime*();  
 Random random = new Random();  
 for (int i = 0; i < count; i++) {  
 int randomNumber = random.nextInt(count);  
 map.put(i, randomNumber);  
 }  
 long endTime = System.*nanoTime*();  
 return (endTime - startTime) / 1\_000\_000;  
 }  
  
 public long getRandomNumbers(int count) {  
 long startTime = System.*nanoTime*();  
 Random random = new Random();  
 for (int i = 0; i < count; i++) {  
 int randomKey = random.nextInt(count);  
 map.get(randomKey);  
 }  
 long endTime = System.*nanoTime*();  
 return (endTime - startTime) / 1\_000\_000;  
 }  
  
 public void shutdown() {  
 hazelcastInstance.shutdown();  
 }  
}

Main.java:

package org.example;  
  
public class Main {  
 public static void main(String[] args) {  
 OracleOps oracleOps = new OracleOps();  
 HazelcastOps hazelcastOps = new HazelcastOps();  
  
 int[] counts = {20000, 100000};  
  
 for (int count : counts) {  
 System.*out*.println("Benchmarking for " + count + " numbers:");  
  
 long oracleInsertTime = oracleOps.insertRandomNumbers(count);  
 System.*out*.println("Oracle Insert time: " + oracleInsertTime + " ms");  
 long oracleSelectTime = oracleOps.selectRandomNumbers();  
 System.*out*.println("Oracle Select time: " + oracleSelectTime + " ms");  
  
 long hazelcastPutTime = hazelcastOps.putRandomNumbers(count);  
 System.*out*.println("Hazelcast Put time: " + hazelcastPutTime + " ms");  
 long hazelcastGetTime = hazelcastOps.getRandomNumbers(count);  
 System.*out*.println("Hazelcast Get time: " + hazelcastGetTime + " ms");  
  
 System.*out*.println();  
 }  
  
 hazelcastOps.shutdown();  
 }  
}

Screenshots:

