**Rule Optimization in Drools**

**Objective:**

This lab exercise will guide you through optimizing rules in Drools to improve performance and maintainability. The exercise will involve creating, testing, and optimizing rules using various techniques.

**Objective**

* Understand how to structure and optimize Drools rules.
* Apply indexing, rule grouping, and agenda management for performance improvements.
* Test the impact of optimizations on rule execution.

**Part 1: Initial Rule Setup**

1. **Create a Simple Drools Project**

Start by setting up a basic Drools project with the following rules:

**rule.drl**:

package com.example.rules;

import com.example.model.Order;

rule "High Value Order"

when

$order: Order(value > 1000)

then

System.out.println("Processing high value order: " + $order.getId());

end

rule "Low Value Order"

when

$order: Order(value <= 1000)

then

System.out.println("Processing low value order: " + $order.getId());

end

**Order.java**:

package com.example.model;

public class Order {

private String id;

private double value;

public Order(String id, double value) {

this.id = id;

this.value = value;

}

public String getId() {

return id;

}

public double getValue() {

return value;

}

}

1. **Test the Initial Setup**

Create a test class to run the Drools session:

**DroolsApp.java**:

package com.example;

import com.example.model.Order;

import org.kie.api.KieServices;

import org.kie.api.runtime.KieContainer;

import org.kie.api.runtime.KieSession;

public class DroolsApp {

public static void main(String[] args) {

KieServices ks = KieServices.Factory.get();

KieContainer kContainer = ks.getKieClasspathContainer();

KieSession kSession = kContainer.newKieSession();

Order order1 = new Order("Order1", 1200);

Order order2 = new Order("Order2", 800);

kSession.insert(order1);

kSession.insert(order2);

kSession.fireAllRules();

kSession.dispose();

}

}

**Run** the application and observe the output to ensure the rules are working as expected.

**Part 2: Rule Optimization**

1. **Rule Indexing**

Modify the rule.drl to apply indexing on the value field. This helps improve the performance when dealing with a large number of Order objects.

**Optimized rule.drl**:

package com.example.rules;

import com.example.model.Order;

rule "High Value Order"

when

$order: Order(value > 1000) @watch(value)

then

System.out.println("Processing high value order: " + $order.getId());

end

rule "Low Value Order"

when

$order: Order(value <= 1000) @watch(value)

then

System.out.println("Processing low value order: " + $order.getId());

end

1. **Using Agenda Groups**

Group the rules into agenda groups to control the order of rule execution.

**Agenda Grouped rule.drl**:

package com.example.rules;

import com.example.model.Order;

rule "High Value Order"

agenda-group "high-value"

when

$order: Order(value > 1000)

then

System.out.println("Processing high value order: " + $order.getId());

end

rule "Low Value Order"

agenda-group "low-value"

when

$order: Order(value <= 1000)

then

System.out.println("Processing low value order: " + $order.getId());

end

**Modify the DroolsApp to Use Agenda Groups**:

package com.example;

import com.example.model.Order;

import org.kie.api.KieServices;

import org.kie.api.runtime.KieContainer;

import org.kie.api.runtime.KieSession;

public class DroolsApp {

public static void main(String[] args) {

KieServices ks = KieServices.Factory.get();

KieContainer kContainer = ks.getKieClasspathContainer();

KieSession kSession = kContainer.newKieSession();

Order order1 = new Order("Order1", 1200);

Order order2 = new Order("Order2", 800);

kSession.getAgenda().getAgendaGroup("high-value").setFocus();

kSession.insert(order1);

kSession.fireAllRules();

kSession.getAgenda().getAgendaGroup("low-value").setFocus();

kSession.insert(order2);

kSession.fireAllRules();

kSession.dispose();

}

}

1. **Measure and Compare Performance**
   * **Add Timing Code**: Measure the time taken to execute rules before and after optimization.
   * **Test with Large Datasets**: Insert a large number of Order objects to observe the impact of optimizations.

long startTime = System.currentTimeMillis();

kSession.fireAllRules();

long endTime = System.currentTimeMillis();

System.out.println("Time taken: " + (endTime - startTime) + "ms");

**Part 3: Advanced Optimization Techniques**

1. **Salience and No-Loop Attributes**

Use the salience attribute to prioritize rule execution and the no-loop attribute to prevent rules from triggering themselves repeatedly.

**Example with Salience and No-Loop**:

rule "High Priority Rule"

salience 10

when

$order: Order(value > 1000)

then

System.out.println("High priority processing for order: " + $order.getId());

end

rule "Low Priority Rule"

salience -10

no-loop true

when

$order: Order(value <= 1000)

then

System.out.println("Low priority processing for order: " + $order.getId());

end

**Submission**

* **Deliverables**: Submit the optimized DroolsApp.java, rule.drl, and any performance metrics you've gathered.
* **Analysis**: Write a brief analysis comparing the performance before and after optimization.