Complex Event Processing

ECA

Implémentation via Drools

Emmanuel Coquery

De quoi parle-t-on ?

- Événement:
  - Message
  - Combinaisons d'événements
- Comment définir des événements complexes : de manière
  - Déclarative
  - Dynamique

Applications

- Monitoring à haut niveau
  - Détection de problèmes complexes
  - Nécessité d'agréger des événements
- Processus métiers dynamiques
  - Workflows changeants
  - Règles métier ayant des combinaisons complexes

Comment faire ?

- Règles d'inférence
  - Utiliser des règles qui:
    - Déduisent des faits/événements
    - À partir d'autres faits/événements
    - Sous certaines conditions
- Requêtes continues (e.g. CQL/STREAM)
  - Requête s'exprimant sur un/des flux de données
  - Produisant un flux de données

Règles *Event Condition Action* (ECA)

- Base de connaissance
  - Base dynamique de faits
  - Mise à jour:
    - Ajout/suppression de faits
  - Par utilisation de règles ECA
  - Moteur d'inférence
    - En général: *forward chaining*

Event

- Déclenche la règle
- Test sur la présence de faits
- Filtrage de motif (*pattern matching*)
  - Contraint la forme
  - Lie les variables
  - Unification unidirectionnelle
- Exemple:
  - Fait: Achat effectués
    - Par un client c
  - Fait: le client c a des points de fidélité

Condition

- Test plus fin
  - Intervient comme un filtre additionnel
- Code externe au moteur d'inférence
  - e.g. test sur la valeur d'un attribut
- Exemple:
  - Le nombre de points de fidélité est > 50

Action

- Mise à jour de la base de faits

- Exécution de code externe
  - Log
  - Ajout de données dans une BD
  - Envoi de message
  - Action système

Exemple (Drools)

```
package com.sample

import com.sample.DroolsTest.Message;

rule "Hello World"
    when
        m : Message( status == Message.HELLO, myMessage : message )
    then
        System.out.println( myMessage );
        m.setMessage( "Goodbye cruel world" );
        m.setStatus( Message.GOODBYE );
        update( m );
end

rule "GoodBye"
    when
        Message( status == Message.GOODBYE, myMessage : message )
    then
        System.out.println( myMessage );
end
```

Exemple 2

```
package tiw5

import tiw5.Achat;
import tiw5.Client;

rule "Ajout Points fidelite"
                        no-loop
    when
        $c : Client ($num : id)
        $a : Achat(id_client == $num)
    then
           System.out.println("Points");
        $c.points += Math.round($a.valeur);
        update( $c );
end
```

```
rule "Reduction"
                        no-loop
                        salience 1
    when
        $c : Client ($num : id, points >= 50)
        $a : Achat(id_client == $num)
    then
           System.out.println("Réduction");
           $a.valeur = $a.valeur * 0.9;
           $c.points -= 50;
           update( $c );
           update( $a );
end
```

Exécution: Rete

- Algorithme efficace pour le déclenchement des règles
  - Concerne la partie *event*
- Basé sur un graphe acyclique
  - Nœuds internes ↔ configurations partielles
    - Gardent une mémoire des configuartion partielles courantes
  - Feuilles ↔ parties gauches (*event*)
- Propagation des nouveaux faits en suivant le graphe

Drools: intégration dans un contexte plus large

- Code Java dans le corps des règles
- ServiceMix:
  - JbiHelper, Exchange (msg)
  - Injection de composants dans les règles
  - Utilisations:
    - Routage, endpoint
- Requêtes en direct (*live queries*):
  - Requête: ensemble des n-uplets de faits
    - Correspondant à l'expression de la requête
    - Vue comme une partie gauche de règle
  - Permet de surveiller l'évolution d'une base de faits
  - *Listener* sur une requête

Corrélations

- Problème similaire aux cas des messages en composition de service

- On utilise plus naturellement des identifiants naturels dans le cas des règles
  - ≈ clé générée vs clé naturelle en BD

Flux d'événements

- Les règles ECA peuvent servir à raisonner sur des flux d'événements
  - Ne pas garder tous les événements en « mémoire »
- Faire disparaître certains événements de la base de faits
  - Raisonnement sur fenêtre temporelle glissante
    - *e.g.* sur les 30 dernière minutes
  - Durée de vie des événements
  - Faire disparaître les événements quand on ne peut plus déduire de faits
    - Difficile à prévoir dans l'absolu
  - Raisonnement sur un nombre d'événement maximal

Aspects temporels

- Corrélation d'événements via le temps:
  - Avant, après, pendant, en même temps, etc
  - Correspondance approximative
- Durée de vie des événements
  - Individuelle
  - Générale
  - Liées aux règles:
    - Si aucune règle ne peut être déclenchée par un événement vieux de plus de 30min, il peut être supprimé
    - Gestion par type d'événements

Fenêtres de temps et agrégation

- Accumulation d'événements
- Similaire au GROUP BY
  - Sur une fenêtre temporelle / # d'événements
- Fonctions d'agrégation
  - Average, sum, …

Exemple (Drools Fusion)

```
rule "Sound the alarm in case temperature rises
above threshold"
when
    TemperatureThreshold( $max : max )
    Number( doubleValue > $max ) from
      accumulate(
         SensorReading( $temp : temperature )
                          over window:time( 10m ),
         average( $temp ) )
then
    // sound the alarm
end
```

Sources

- Drools: http://www.jboss.org/drools
- Wikipedia:
  - CEP: http://en.wikipedia.org/wiki/Complex_event_processing
  - Rete: http://fr.wikipedia.org/wiki/Algorithme_de_Rete