Replay the Execution History of Rule-based Information

Essam Mansour

March 2009
Outline

1. Introduction
   - Background
   - Research Problem
   - Research Objective

2. Contributions
   - DRDoc Model
   - AIMQL Language
   - AIMS System
   - Evaluation

3. Conclusion

4. Contact Me
Applications, such as disease management or customer relationship management, monitor changes of interest.

Information formalizing this monitoring process could be modeled as ECA rules (Rule-Based Information).

Approaches adopting active database implement this information as triggers.
In healthcare, clinical guidelines are used to monitor the clinical events of interest concerning a specific patient.

a medical patient plan is generated based on a clinical guideline for each diabetes patient.

In the execution process, medical patient plan determines the reaction to the changes of the patients state.

Clinicians are interested to review the execution history of these medical plans.
We developed a framework, called SIM, for managing rule-based information using XML and active databases. The management process goes through three phases:

- **Specification**: the rule-based information (such as clinical guidelines) is formalized at a generic form.
- **Instantiation**: an entity specific plan (such as medical patient plan) is generated from the generic form.
- **Maintenance**: the plans are executed, manipulated and queried.
Execution History contains several information scenes. End users, such as clinicians, are interested to:

- replay the execution history of a specific plan,
- determine the action made, when and why,
- compare the progress of two plans, and
- specify how many times a rule was executed
Move the complexity of querying the rule-based information and its execution history from user/application code to a high level declarative language.
Research Objective

**Objective Statement**

Develop a replay support management that is based on XML technologies and DBMS.

The main aspects of the replay support management are:

- a **model** for formalizing the plans generated for a specific entity, such as patient, and
- a **replay query language** that declaratively query the plans and their execution history.
stands for Dynamic Rule-based Document

Formalizes the plans generated for specific entities and their history as a temporal XML document, called DRDoc document.

DRDoc document consists of two main parts, active and passive.
DRDoc: A model for the Rule-Based Information

DRDoc document

Active Part
- the reactive behavior of the rule-based information
- coded as several triggers.

Passive Part
- descriptive information, which represents
  - states of the rule-based information and
  - the execution history
DRDoc: A model for the Rule-Based Information

The XML Schema of DRDoc.
An Example

Part of a medical plan

Rule MAP1

**ON** 2 days **After the patient admission**,  
**DO** order the test albumin creatine ratio (ACR).

Rule MAP2

**ON** receiving the result of the ACR test,  
**IF** the result **is greater than 25**  
**DO** Add rule MAP3.
An Example

A part of a patient plan..
- AIMQL plays over again the history of DRDoc documents to show the in details the actions that cause changes on them.
- The AIMQL replay query statement consists of main three clauses as follows:

  - **REPLAY** Clause: indicates the element that is subject to be replayed
  - **SHOW** Clause: determines which pieces of information are to be returned
  - **WHERE** Clause: includes a comparison predicate
AIMQL Replay Language

The AIMQL replay query structure
DRDoc: A model for the Rule-Based Information

Examples

Replay plan \((X, PID)\) over the period \((TP1, TP2)\)

```
REPLAY PLAN p1
SHOW When OF p1
WHERE p1[DEID = X and SPID = PID]
    and p1.overlaps(valid(TP1,TP2))
```

Retrieve the first instance of the plan \((X, PID)\)

```
REPLAY PLAN p1
SHOW When OF FIRST(p1)
WHERE p1[DEID = X and SPID = PID]
```
Examples

Compare the progress of plan \((X1, PID1)\) with plan \((X2, PID2)\) after its state \((ST)\)

```
REPLAY PLAN p1,p2
SHOW When, How, Why OF p1
WHERE p1[@DEID = X1 and @SPID = PID1] and
  p2[@DEID = X2 and @SPID = PID2] and
NOT (p1.precedes(valid(p2.state[value=ST])))
```
AIMS: A Proof-of-concept System

- stands for **Advanced Information Management System**
- utilizes the modern DBSs that provide XML data management and triggering mechanism
- extends the XML support provided by DBSs to provide temporal support and
- utilizes the DBS to validate and store the DRDOC documents
- maps the AIMQL queries into XQuery scripts
AIMS: A Complex Information Management System

- Specification
- Instantiation
- Execution
- Queries and Manipulations
- Skeletal Plan Doc
- Entity-Specific Plan Doc

- Temporal Execution
- Rule Manipulation

- Validation method
- Temporal storage method
- Temporal query method

- Dissemination method
- Distribution method

- Domain Information
- AIMSL Specification
- AIM ESPDoc

- Registration and Manipulation
- Modifications

- Specification
- Manipulation
- Query and Replay

- Result and Acknowledgement

AIMS: A Proof-of-concept System

Essam Mansour

DBKDA 2009 Cancun, Mexico
AIMS: A Proof-of-concept System

The XQuery script of the replay query “retrieve the first instance of the plan (X,PID)”

```xquery
declare namespace xsd = "http://www.w3.org/2001/XMLSchema";
for $p1 in
db2-fn:xmlcolumn('AIM_ESPlan_TAB.ESPDOC')//Plan[@DEID=X and @SPID=PID]
where(
  ( xsd:dateTime($p1/@startTime) = xsd:dateTime($p1/@startTime) )
)
return
<Plan domainEntity_ID="{$p1/@DEID}" protocol_ID="{$p1/@SPID}'"
  startTime="{$p1/@startTime}" endTime="{$p1/@endTime}'">
  { for $PState in $p1/state
    return
      <state startTime="{$PState/@startTime}" endTime="{$PState/@endTime}'">
        { for $PState/value[( ( xsd:dateTime($p1/@startTime) = xsd:dateTime($PState/@endTime) )
            and ( xsd:dateTime($PState/@startTime) = xsd:dateTime($p1/@startTime) )
        )] } </state> }
  { for $PSches in $p1/schedules
    return
      <schedules startTime="{$PSches/@startTime}" endTime="{$PSches/@endTime}'">
        { for $sch in $PSches/schedule
          where( ( xsd:dateTime($sch/@startTime) = xsd:dateTime($p1/@startTime) )
          )
          return
            <schedule IDREF="{$sch/@IDREF}'", startTime="{$sch/@startTime}" endTime="{$sch/@endTime}'">
              { for $sch/scheduleRules/rule
                where( ( xsd:dateTime($rul/@startTime) = xsd:dateTime($sch/@startTime) )
                )
                return
                  <rule IDREF="{$rul/@IDREF}'", startTime="{$rul/@startTime}" endTime="{$rul/@endTime}'">
                    { for $RState in $sch/state
                      where( ( xsd:dateTime($sch/@startTime) = xsd:dateTime($RState/@startTime) )
                      )
                      return
                        <state startTime="{$RState/@startTime}" endTime="{$RState/@endTime}'">
                          { for $RState/value[( ( xsd:dateTime($p1/@startTime) = xsd:dateTime($RState/@endTime) )
                            and ( xsd:dateTime($RState/@startTime) = xsd:dateTime($p1/@startTime) )
                          )] } </state> }
                  </rule> }
            </schedule>
        </schedules>
      </schedules>
  </schedules>
</Plan>
```
Experiments are tested on

- Debian 4, a Linux system
- an Intel Pentium III processor machine
- one Gigabyte RAM and
- 40 Gigabyte hard disk
Evaluation: The storage efficiency

The DRDoc document size vs. number of updates.
Evaluation: The Query Performance

The query elapsed time (millisecond) vs. the DRDoc document size (KB)

- The query elapsed time increases linearly with the DRDoc document size.
- The relationship can be modeled using a linear regression.

Essam Mansour
DBKDA 2009 Cancun, Mexico
March 2009
The DRDoc model formalizes the rule-based information and its execution history as one temporal XML document.

The AIMQL language queries the rule-based information and its execution history at a declarative level.

AIMS maps the AIMQL replay queries into XQuery scripts that are to be executed by the DBS.

The evaluation of the AIMS storage and query performance shows positive results.
Future work

- doing additional experiments with different workloads and query sets
- developing advanced graphical visualization mechanism to review the replayed information
- developing a method that provides automatic discovery of information from the execution history
Thank You

Dr. Essam Mansour
Research Associate
School of Information Technology
International University in Germany
Email: essam.mansour@ieee.org
Web page: http://it.i-u.de/dbis