Поддержка триггеров в системах XML-баз данных

Мария Гринева
rekouts@ispras.ru

Институт Системного Программирования РАН

План доклада

- Применения триггеров
- Общий вид триггеров
- Основные принципы реализации активных правил в РСУБД POSTGRES
- Триггеры в XML-базах данных
- Проблемы реализации триггеров в системах XML-баз данных:
  - Методы выявления активируемых триггеров
  - Отслеживание корректности вложенных update-операций

Системы, построенные на использовании триггеров

- Экспертные системы, обрабатывающие большие объемы данных (*data-intensive expert systems*)
- Системы управления технологическим потоком (*workflow management systems*)

Задачи, решаемые при помощи триггеров

- Поддержка ограничений целостности
- Авторизация
- Сбор статистики
- Поддержка представлений

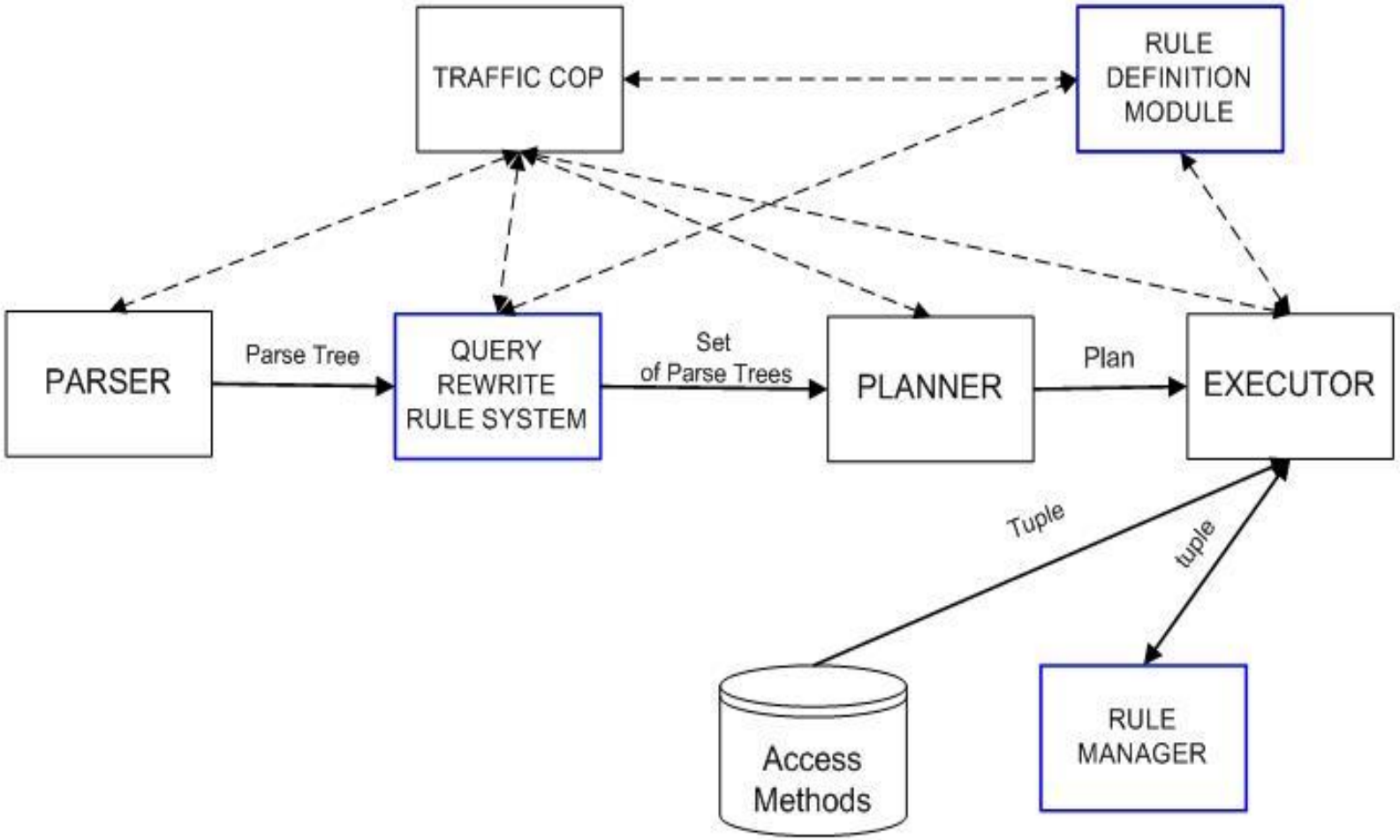
Общая форма триггеров

- Определяются пользователями, приложениями, администраторами базы данных
- Общая форма состоит из трех частей:
  - Событие: вызывающее срабатывание триггера
  - Условие: проверяется при срабатывание триггера
  - Действие: выполняется при запуске триггера, если выполнено его условие

Принципы реализации активных правил в РСУБД POSTGRES

- Поддержка активных правил на уровне кортежей (Tuple Level System)
- Поддержка посредством переписывания запросов (Query Rewrite System)

Общая архитектура системы



Поддержка на уровне кортежей (Tuple Level System)

- Кортежи помечаются специальными блокировками правил (rule lock):
  - Имя правила
  - Тип блокировки (кортежа или всего отношения)
  - Идентификатор плана правила
- Правильность блокировок правил при дальнейших модификациях данных поддерживается при помощи rule stubs

Поддержка правил посредством переписывания запросов

Правило:

```
define rule r1 is
on retrieve to EMP.salary
where CURRENT.name = ‘Sam’
do instead
    retrieve(salary=3*EMP.salary)
            where EMP.name=‘Bill’
```

Запрос:

```
retrieve (EMP.salary)
        where EMP.age > 30
```

Результат переписывания:

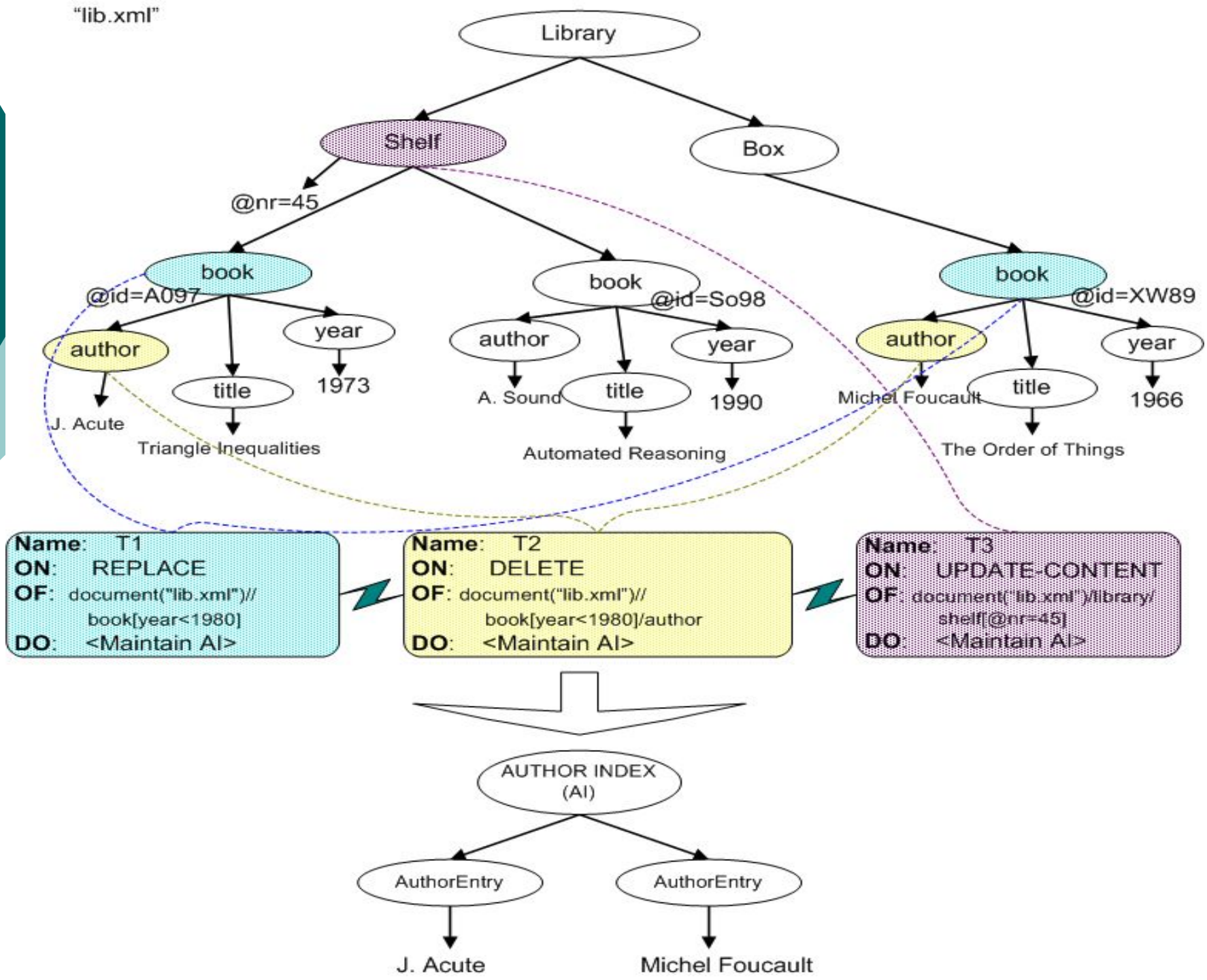
```
retrieve (EMP.salary)
where EMP.age > 30 and
    not(EMP.name = ‘Sam’)
```

```
retrieve (salary = 3*E.salary)
from E in EP
where EMP.age > 30
 and EMP.name=‘Sam’ and E.name=‘Bill’
```

M. Stonebraker
“On rules, procedures, caching and views in data base systems”

Выбор оптимальной подсистемы в POSTGRES

- Правило покрывает небольшое кол-во кортежей => TLS эффективней
- Если правило – определение view над существующим отношением => QLS эффективней


"lib.xml"
Library
Shelf
Box
@nr=45
book
@id=A097
author
J. Acute
title
Triangle Inequalities
year
1973
book
@id=So98
author
A. Sound
title
Automated Reasoning
year
1990
book
@id=XW89
author
Michel Foucault
title
The Order of Things
year
1966
Name: T1
ON: REPLACE
OF: document("lib.xml")//
book[year<1980]
DO: <Maintain AI>
Name: T2
ON: DELETE
OF: document("lib.xml")//
book[year<1980]/author
DO: <Maintain AI>
Name: T3
ON: UPDATE-CONTENT
OF: document("lib.xml")/library/
shelf[@nr=45]
DO: <Maintain AI>
AUTHOR INDEX
(AI)
AuthorEntry
AuthorEntry
J. Acute
Michel Foucault

Синтаксис и семантика XML-триггеров

CREATE TRIGGER trigger-name

(BEFORE|AFTER)

(INSERT|DELETE|REPLACE|UPDATE-CONTENT)+

OF XPathExpression (,XPathExpression)*

[FOR EACH (NODE|STATEMENT)]

[WHEN XQuery-Clause]

DO [INSTEAD] XQuery-UpdateOp

Проблемы реализации триггеров в системах XML-баз данных

- Выявление триггеров, активируемых данной update-операцией
- Отслеживание корректности update-операций, вызываемых в качестве действия триггера (проблема вложенных update-операций)
- Каскадный запуск триггеров
- Оптимизация вычисления условий триггеров

Методы выявления триггеров, активируемых данной update-операцией: **Требования к системе**

- Поддержка XQuery и update-языка, основанного на модели данных XQuery
- Поддержка операций сравнения узлов в отношении предок-потомок
- Поддержка *описывающей схемы* (Data Guide)

Методы выявления триггеров, активируемых данной update-операцией: **Универсальный метод**

- Вычисление update-пути и триггер-путей
- Сравнение полученных последовательностей узлов посредством операций отношения предок-потомок
- Выбор активируемых триггеров по таблице (на следующем слайде)

Методы выявления триггеров, активируемых данной update-операцией:

Универсальный метод

		update operation		
		Delete	**Insert**	**Replace**
update vs. trigger path length	length(tr_pth)>length(up_pth)	Delete	Insert	Delete, Insert
	length(tr_pth)=length(up_pth)	Delete	Insert	Delete, Replace
	length(up_pth)-length(tr_pth) = 1	Update-content	Update-content	Update-content, Insert
	length(up_pth)-length(tr_pth) >1	Update-content	Update-content	Update-Content

Методы выявления триггеров, активируемых данной update-операцией: **Оптимизированный Метод**

- **1-ый этап** на стадии компиляции update-операции: сужение набора триггеров, *возможно* активируемых данной операцией модификации при помощи описывающей схемы
- **2-ой этап** на стадии выполнения update-операции: обработка триггеров <u>встраивается</u> в план выполнения update-операции

Оптимизированный метод:
1-ый этап – выбор возможных триггеров на стадии компиляции

- Перепишем update-путь и триггер-пути в абсолютные пути: то есть исключим неопределенности '//','*', '..'
- Выберем возможные триггеры: сравним имена элементов на соответствующих шагах триггер-пути и update-пути
- Выбор возможных триггеров по таблице

example:

UPDATE replace
doc("lib.xml")/library/shelf[@nr=45]/book[@id="A097"]
as $b with <book id=… </book>
T1: [ON Replace]; doc("lib.xml")//book[year<1980]

---

update-path=doc("lib.xml")/library/shelf[@nr=45]/book[@id="A097"]
T1-path=doc("lib.xml")/library/shelf/book[year<1980]

Оптимизированный метод: выбор возможных триггеров по таблице

		update operation		
		Delete	**Insert**	**Replace**
update vs. trigger path length	length(tr_pth)>length(up_pth)	Delete	Insert	Delete, Insert
	length(tr_pth)=length(up_pth)	Delete	Insert	Delete, Replace
	length(up_pth)-length(tr_pth) = 1	Update-content	Update-content	Update-content, Insert
	length(up_pth)-length(tr_pth) >1	Update-content	Update-content	Update-Content

Оптимизированный метод:
2-ый этап – выстраивание обработки триггеров в план выполнения update-операции

«*Объединенный план выполнения*» строится таким образом, что предикаты из триггер-путей встраиваются в план вычисления update-пути посредством mark-операций

- mark1(unmarked_seq, predicate, trigger_name)
- mark2(marked_seq, predicate, trigger_name)

example:

(select (mark1(child(select(child (child (doc("lib.xml"),elem(library)), elem(shelf)), f(s|attr(s,nr)=45)), elem(book)), f(a|child(a,year)<1980),"T1"), f(b|attr(b,id))="A097"))

Оптимизированный метод: результат

В результате выполнения «объединенного плана»

- Определены триггеры, активируемые данной update-операцией
- Определен набор узлов, на которых выявленные триггеры должны сработать

Отслеживание корректности update-операций, вызываемых в действии триггеров

- Действие триггера – update-операция => может затрагивать узлы, модифицируемые внешней update-операцией (вызвавшей триггер)
- Решение – использование транзакционных блокировок

Заключение

Предложенные методы разработаны в рамках проекта по реализации XML-СУБД SEDNA

http://www.modis.ispras.ru/Development/sedna.htm

Список статей

The POSTGRES Rule System. *M. Stonebraker, S. Potamianos*

- Introducing Trigger Support for XML Database Systems, *M. Rekouts, M. Grinev,* in Proc of SYRCoDIS-2005
- Incorporating Active Rules Processing into Update Execution in XML Database Systems, *M. Rekouts*, in Proc of DEXA Workshops 2005

http://www.modis.ispras.ru/publications.htm

rekouts@ispras.ru