

Query Rewriting for Rule Mining in Databases



Brice Chardin
INSA/Université de Lyon, CNRS

Marie Pailloux
Université Blaise Pascal, CNRS

Emmanuel Coquery
Université Lyon 1/Université de
Lyon, CNRS

Jean-Marc Petit
INSA/Université de Lyon, CNRS

Benjamin Gouriou
Université Blaise Pascal

Work partially funded by the French Research Agency (ANR), DAG project




Motivation

- **Context :**
 - Relational databases everywhere
 - RDBMS market expected to double by 2016 [MarketResearch.com, Aug 2012]
 - **LML 2013**
 - Languages for **data mining and machine learning**
 - Languages for **databases**
 - Algebra, **TRC**, DRC, Datalog, QBE and **SQL**
- **Simple goals :**
 - Query the data where they are
 - Use/extend DB languages for pattern mining problems (e.g. DMQL, MSQL, ...)

Pattern mining ↔ query processing

- **Fact** : polynomial operators in DB and quasi-polynomial (or exponential) operators in data mining
- **Intuition**: universal language is a dream, specific languages for some class of patterns a reality
- **Our choice**: patterns = **RULES complying with Armstrong axioms, i.e. logical implication**
- **Proposition**:
 - **SafeRL**: a well-founded logical query language derived from **TRC**
 - **RQL**: Its practical counterpart derived from **SQL**

- 
- Motivating examples
 - Contributions
 - SafeRL
 - Query rewriting
 - RQL syntax
 - RQL Web Application
 - Conclusions

Outline

Three examples of rules

- 1st example: Implications in FCA (or association rules with 100% confidence)
 - Satisfaction in a $\{0,1\}$ -relation r
 - $\forall t \in r, \text{ if } \forall A \in X, t[A]=1 \text{ then } \forall B \in Y, t[B]=1$
- 2nd example: Functional dependencies in DB
 - Satisfaction in a n -ary relation r
 - $\forall t_1, t_2 \in r, \text{ if } \forall A \in X, t_1[A]=t_2[A] \text{ then } \forall B \in Y, t_1[B]=t_2[B]$
- **Remark**
 - Same syntax, different meaning
 - Same reasoning on rules for both
 - [Armstrong axioms](#)
 - [Closure system, covers ...](#)

Another kind of rules (1/2)

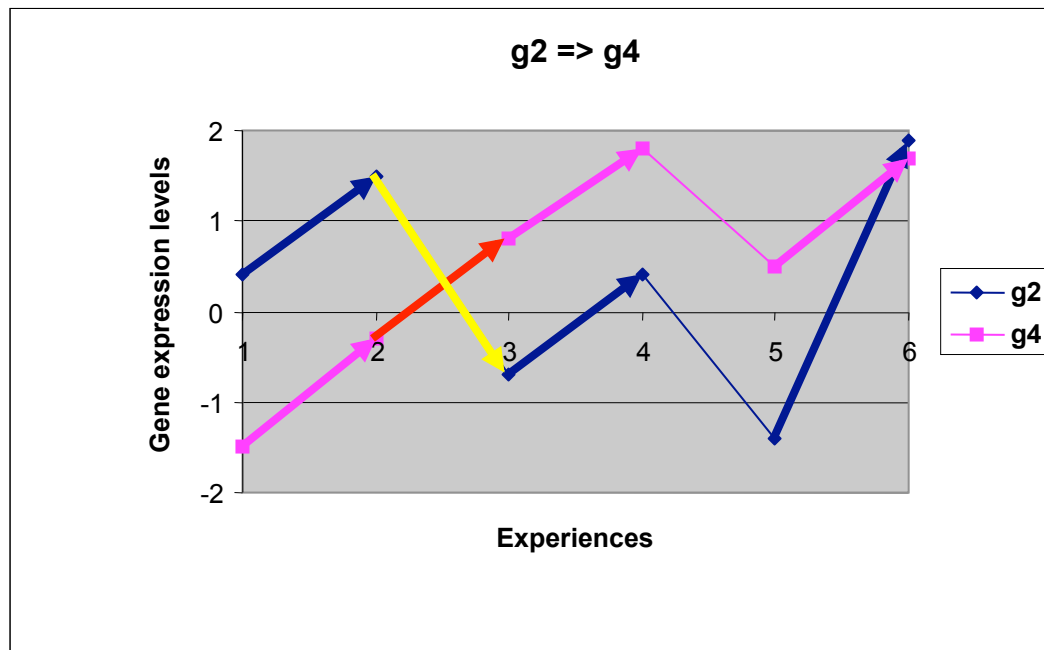
- ➔ Example from gene expression data
- ➔ **Assume tuples are ordered**
- ➔ **Idea:** adapting FD for catching the evolution of attributes between two consecutive tuples

	g1	g2	g3	g4	g5	g6	g7	g8
t1	1,9	0,4	1,4	-1,5	0,3	1,8	0,8	-1,4
t2	1,7	1,5	1,2	-0,3	1,4	1,6	0,7	0,0
t3	1,8	-0,7	1,3	0,8	-0,1	1,7	0,9	0,6
t4	-1,8	0,4	1,7	1,8	0,6	-0,4	1,0	1,5
t5	-1,7	-1,4	0,9	0,5	-1,8	-0,2	1,2	0,3
t6	0,0	1,9	-1,9	1,7	1,7	-0,5	1,1	1,3

$X \Rightarrow Y$ is **satisfied** in r iff $\forall t_i, t_{i+1} \in r$,
if $\forall g \in X, t_{i+1}[g] - t_i[g] \geq \varepsilon_1$ then $\forall g \in Y, t_{i+1}[g] - t_i[g] \geq \varepsilon_1$
 $\varepsilon_1 = 1.0$

Another kind of rules (2/2)

	g1	g2	g3	g4	g5	g6	g7	g8
t1	1,9	0,4	1,4	-1,5	0,3	1,8	0,8	-1,4
t2	1,7	1,5	1,2	-0,3	1,4	1,6	0,7	0,0
t3	1,8	-0,7	1,3	0,8	-0,1	1,7	0,9	0,6
t4	-1,8	0,4	1,7	1,8	0,6	-0,4	1,0	1,5
t5	-1,7	-1,4	0,9	0,5	-1,8	-0,2	1,2	0,3
t6	0,0	1,9	-1,9	1,7	1,7	-0,5	1,1	1,3



$$r \models g_2 \Rightarrow g_4$$

If expression level of g_2 grows
between t_i and t_{i+1} ,
then expression level of g_4
also grows

But $r \not\models g_4 \Rightarrow g_2$

t_2, t_3 is a counter-example

Ingredient for the semantics of rules

- Number of tuples ?
 - 1 for implication, 2 for FD and variants
 - In fact, could be 3, 4 ...
- Tuple defined over what ?
 - the result of a SQL statement
- Condition to be verified?
 - Classical logical formulas on tuples and every attribute of LHS+RHS



- Motivating examples

- Contributions

- SafeRL

- Query rewriting

- RQL syntax

- RQL Web Application

- Conclusions



Outline

SafeRL: a query language for rules

- SafeRL: a well-founded logical query language

$$Q = \{X \rightarrow Y \mid \forall t_1 \dots \forall t_n (\psi(t_1, \dots, t_n) \wedge (\forall A \in X (\delta(A, t_1, \dots, t_n)) \rightarrow \forall A \in Y (\delta(A, t_1, \dots, t_n))))\}$$

- Syntax + semantics **not detailed here**: cf paper
- Every SafeRL query Q defines rules “equivalent to” FD or implications
- **Result**: There is a **closure system $C(Q)$** associated to Q

Contribution: Query rewriting

➤ A **base B** of a closure system C is such that

$$\text{Irreducible}(C) \subseteq B \subseteq C$$

➤ **Main result:**

Let Q be a SafeRL query over a DB d

THM : There exists a *SQL query* Q' over d such that Q' computes a **base B** of C(Q), the closure system associated to Q

Base of a query: the data-centric step

From the **base B** of Q in d, we can get:

- The **closure** of an attribute set
- The **canonical cover** of satisfied rules
- The cover of Gotlob&Libkin of **approximate rules**

and we can decide whether or not a given rule is satisfied

- If not, a **counter example** from d can be provided
- *Nothing new here*, cf related works

RQL syntax

➤ RQL: A practical query language for SafeRL

➤ Look and feel similar to SQL

FINDRULES

OVER A_1, \dots, A_n

SCOPE $t_1 (SQL_1), \dots, t_n (SQL_n)$

WHERE COND(t_1, \dots, t_n)

CONDITION ON A IS COND' (A, t_1, \dots, t_n)

RQL examples

- Let **DEPT** and **EMP** be two relations
 - **DEPT**(DEPTNO, DEPTNAME, MGRNO, ADMRDEPT, LOC)
 - **EMP**(EMPNO, LASTNAME, WORKDEPT, JOB, EDUCLEVEL, SEX, SAL, BONUS, COMM, MGRNO)

- Rules between NULL values in **DEPT** ?
 - FINDRULES
 - OVER Deptname, Mgrno, Admrdept, Loc
 - SCOPE t1 DEPT
 - CONDITION ON A IS t1.A IS NULL

Cont'ed

➔ FD satisfied in **EMP**

```
FINDRULES
```

```
OVER Empno, Lastname, Workdept, Job, Sex, Bonus
```

```
SCOPE t1, t2 Emp
```

```
CONDITION ON A IS t1.A = t2.A
```

Rules over genes

$X \Rightarrow Y$ is **satisfied** in r iff $\forall t_i, t_{i+1} \in r$,
if $\forall g \in X, t_{i+1}[g] - t_i[g] \geq \varepsilon_1$ **then** $\forall g \in Y, t_{i+1}[g] - t_i[g] \geq \varepsilon_1$
 $\varepsilon_1 = 1.0$

FINDRULES

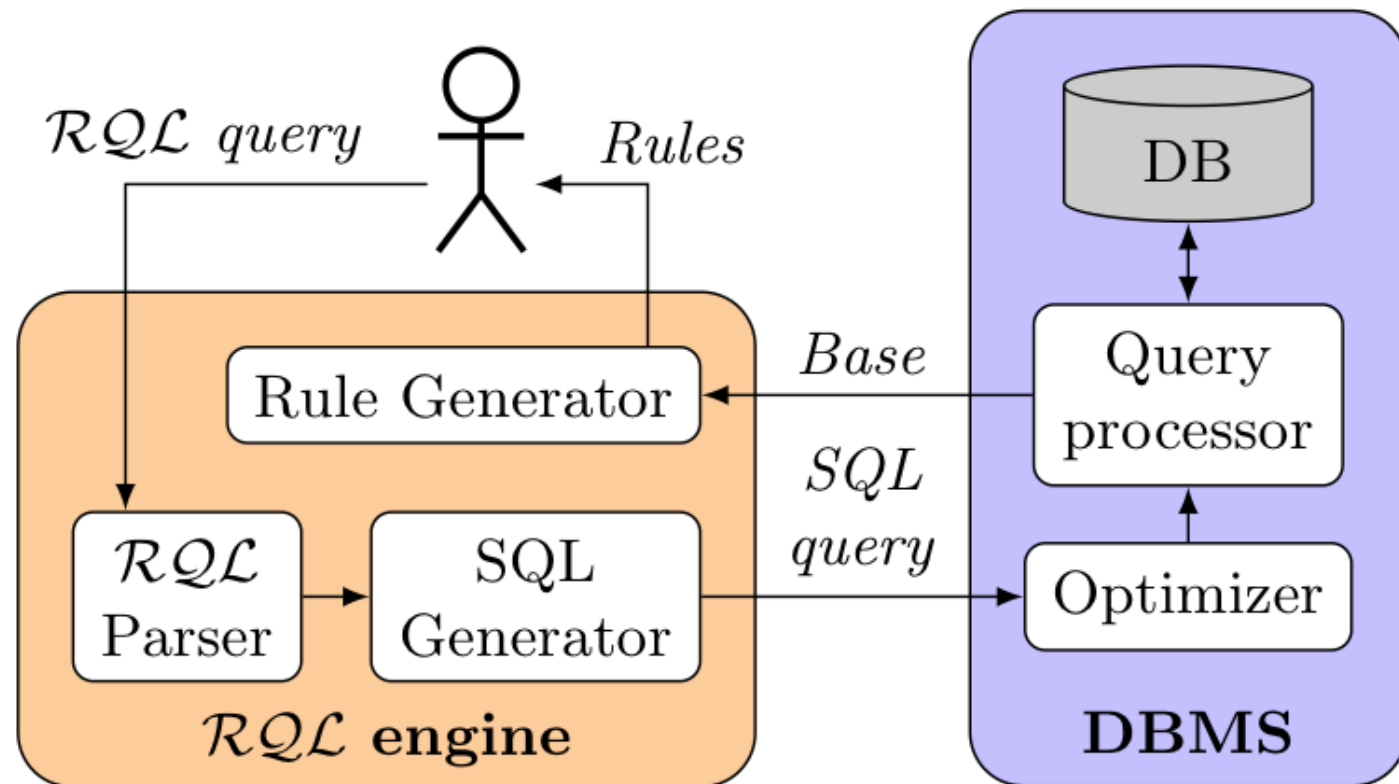
OVER $g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8$

SCOPE t_1, t_2 GENES

WHERE $t_2.sample = t_1.sample + 1$

CONDITION ON A IS $t_2.A - t_1.A \geq 1.0$

Architecture



RQL Web application

- Open to registered users (simple application form)
- Web Framework For Java
 - Play Framework (<http://www.playframework.com/>)
 - DBMS: Oracle v11 (+ MySQL)
 - + specific development in C++, Java, C (Uno's code)
- Two modes: **Sample** (Guest user) and **SandBox** (dedicated Oracle user, 200Ko quota)
 - SQL + RQL
- RQL Web App : <http://rql.insa-lyon.fr>

-
- Motivating examples
 - Contributions
 - SafeRL
 - Query rewriting
 - RQL syntax
 - RQL Web Application
 - Conclusions

Outline

Conclusion

- From a logical query language for rules to the practical language RQL
 - Easy to use by SQL-aware analysts
 - No discretization
 - Promoting query processing techniques in pattern mining
- RQL: a practical Web application
 - For teaching
 - For research
- A step towards **declarative pattern mining techniques**

**Merci !
Questions ?**

Work partially supported by the French
ANR project DAG (ANR DEFIS 2009)

