Secure XML Database Access with Views

SecReT'09

Benoit Groz
(joint work with Anne-Cécile Caron,Yves Roos, Sławek Staworko, Sophie Tison)

Mostrare

10 juillet 2009

Securing databases with views

Many ways to enforce access control for XML. Among others:

- Checking the queries:
  - statically$\Rightarrow$ may reject proper queries and access [Oasis project: XACML]
  - dynamically$\Rightarrow$ incurs costly runtime security check [Murata et al. CCS'03]

Securing databases with views

Many ways to enforce access control for XML. Among others:

- Checking the queries:
  - statically $\Rightarrow$ may reject proper queries and access [Oasis project: XACML]
  - dynamically $\Rightarrow$ incurs costly runtime security check [Murata et al. CCS'03]
- Annotating the data:
  - annotating the data, or materializing the view $\Rightarrow$ expensive maintenance [Damiani et al. EDBT'00, Cho et al. VLDB'02]
  - **annotating the DTD with *Non-materialized view***
    Rewriting queries from the view to the document
    [Fan et al. SIGMOD'04, Vercammen et al, Rassadko et al ...]

Outline

1. Non-materialized views and query rewriting

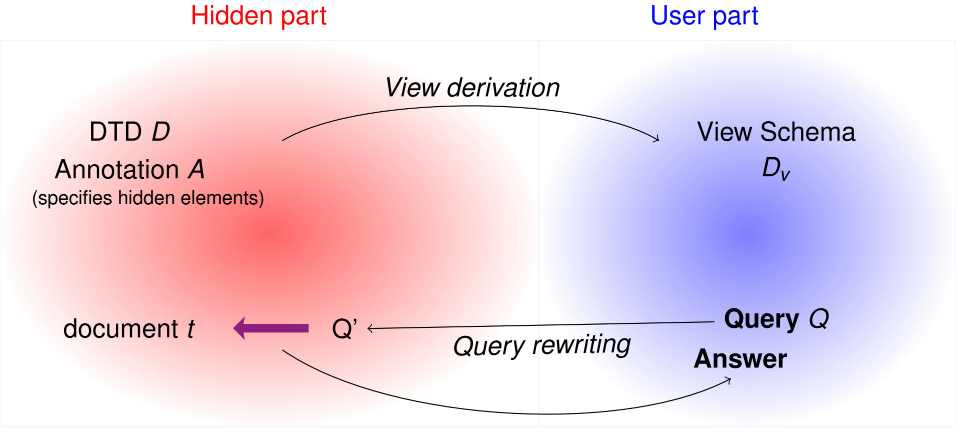
2. Comparing Access Control Policies

Visibilium omnium... et invisibilium

"Whoever wishes to keep a secret
must hide the fact that he possesses one".

attributed to Johann Wolfgang von Goethe

Overview

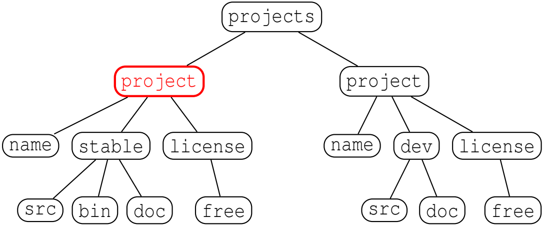


Answer to query Q = evaluation of Q' on the original document t

Framework: XML

- XML document=tree.
- No data-values.

<projects>
<project>
<name>
</name>
...
</project>
<project>
...
</project>
</projects>

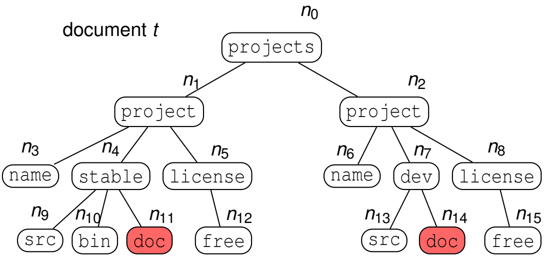


RegularXPath

- We use *Regular XPath* queries

Query $q_1 = \Downarrow^* / \Downarrow :: doc$

$\mathrm{Ans}(q_1, t) = \{n_{11}, n_{14}\}$



“get all documentations”

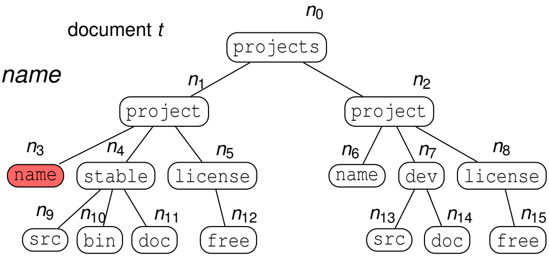
RegularXPath

- We use *Regular XPath* queries

Query $q_2 =$

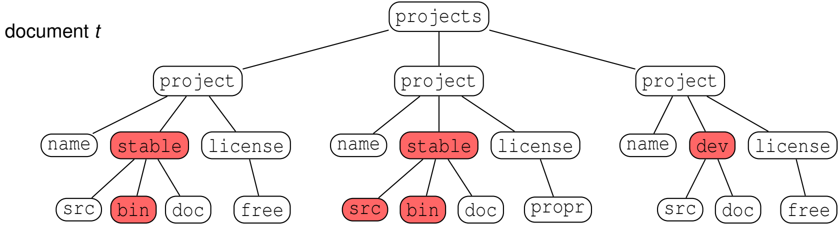
$$\Downarrow :: project[\Downarrow :: stable]/\Downarrow :: name$$

$\mathrm{Ans}(q_2, t) = \{n_3\}$



"get names of stable projects"

Access control for XML

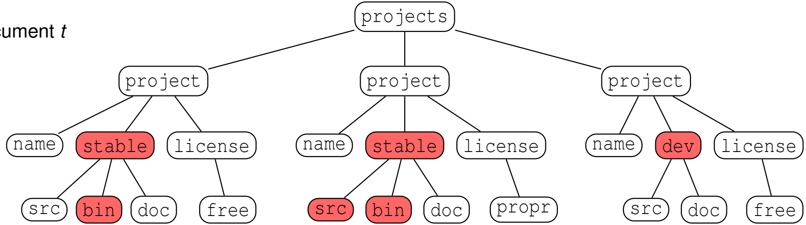


We wish to hide:

- whether a project is *stable* or *in-development*
- the *binaries*
- the *sources* for non-free projects

DTD and Annotation

document t



Example

$\texttt{projects} \rightarrow \texttt{project}^*$

$\texttt{project} \rightarrow \texttt{name}, (\texttt{stable} \mid \texttt{dev}), \texttt{license}$

$\quad A_0(\texttt{project}, \texttt{stable}) = \text{false}$

$\quad A_0(\texttt{project}, \texttt{dev}) = \text{false}$

$\texttt{license} \rightarrow \texttt{free} \mid \texttt{propr}$

$\texttt{stable} \rightarrow \texttt{src}, \texttt{bin}, \texttt{doc}$

$\quad A_0(\texttt{stable}, \texttt{src}) = [\Uparrow^*::\texttt{project}/\Downarrow^*::\texttt{free}]$

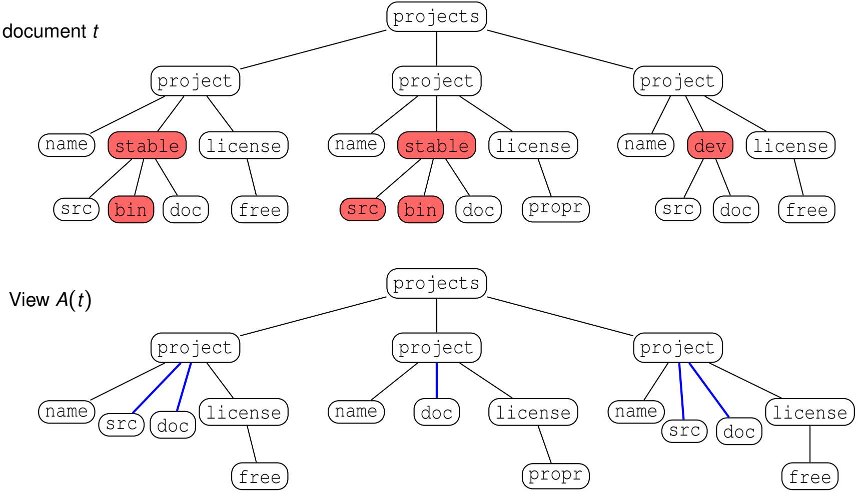
$\quad A_0(\texttt{stable}, \texttt{doc}) = \text{true}$

$\texttt{dev} \rightarrow \texttt{src}, \texttt{doc}$

$\quad A_0(\texttt{dev}, \texttt{src}) = [\Uparrow^*::\texttt{project}/\Downarrow^*::\texttt{free}]$

$\quad A_0(\texttt{dev}, \texttt{doc}) = \text{true}$

The security view



Annotating the DTDs

▷ annotation as a function $A : \Sigma \times \Sigma \longrightarrow \{\text{true}, \text{false}, [f]\}$.

Example

$\texttt{projects} \rightarrow \texttt{project}^*$
$\texttt{project} \rightarrow \texttt{name}, (\texttt{stable} \mid \texttt{dev}), \texttt{license}$
$\quad A_0(\texttt{project}, \texttt{stable}) = \text{false}$
$\quad A_0(\texttt{project}, \texttt{dev}) = \text{false}$
$\texttt{license} \rightarrow \texttt{free} \mid \texttt{propr}$

$\texttt{stable} \rightarrow \texttt{src}, \texttt{bin}, \texttt{doc}$
$\quad A_0(\texttt{stable}, \texttt{src}) = [\Uparrow^*{::}\texttt{project}/\Downarrow^*{::}\texttt{free}]$
$\quad A_0(\texttt{stable}, \texttt{doc}) = \text{true}$
$\texttt{dev} \rightarrow \texttt{src}, \texttt{doc}$
$\quad A_0(\texttt{dev}, \texttt{src}) = [\Uparrow^*{::}\texttt{project}/\Downarrow^*{::}\texttt{free}]$
$\quad A_0(\texttt{dev}, \texttt{doc}) = \text{true}$

Annotating the DTDs

▷ annotation as a function $A : \Sigma \times \Sigma \rightarrow \{\text{true}, \text{false}, [f]\}$.

Example

$\texttt{projects} \rightarrow \texttt{project}^*$
$\texttt{project} \rightarrow \texttt{name}, (\texttt{stable} \mid \texttt{dev}), \texttt{license}$
$A_0(\texttt{project}, \texttt{stable}) = \text{false}$
$A_0(\texttt{project}, \texttt{dev}) = \text{false}$
$\texttt{license} \rightarrow \texttt{free} \mid \texttt{propr}$

$\texttt{stable} \rightarrow \texttt{src}, \texttt{bin}, \texttt{doc}$
$A_0(\texttt{stable}, \texttt{src}) = [\Uparrow^*::\texttt{project}/\Downarrow^*::\texttt{free}]$
$A_0(\texttt{stable}, \texttt{doc}) = \text{true}$
$\texttt{dev} \rightarrow \texttt{src}, \texttt{doc}$
$A_0(\texttt{dev}, \texttt{src}) = [\Uparrow^*::\texttt{project}/\Downarrow^*::\texttt{free}]$
$A_0(\texttt{dev}, \texttt{doc}) = \text{true}$

Proposition

This model of annotation is equivalent to defining accessible elements with a $\mathcal{X}Reg$ filter f^A_{acc} such that :
$\forall n \in N_t.\ n$ accessible wrt. $A \iff (t, n) \models f^A_{acc}$

Rewriting Queries

Theorem: *Regular XPath* **is closed under query rewriting**

There exists a function Rewrite such that :

$$\forall t.\ Ans(Q, A(t)) = Ans(\text{Rewrite}(Q, A), t)$$

Moreover, $\text{Rewrite}(Q, A)$ is computable in time $O(|A| * |Q|)$.

Rewriting Queries

Theorem: *Regular XPath* **is closed under query rewriting**

There exists a function Rewrite such that :

$$\forall t.\ Ans(Q, A(t)) = Ans(\text{Rewrite}(Q, A), t)$$

Moreover, $\text{Rewrite}(Q, A)$ is computable in time $O(|A| * |Q|)$.

Proof.

Translate the base axes using f^A_{acc}:

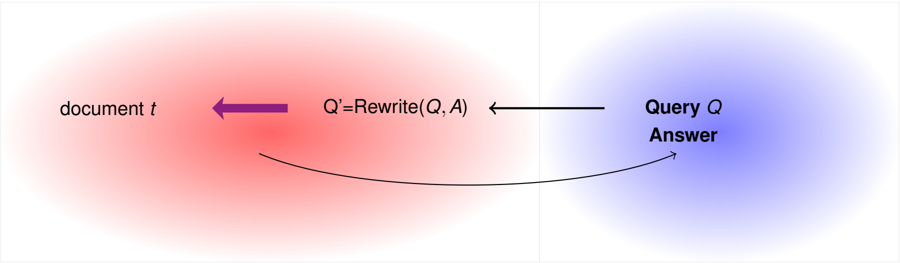
$\text{Rewrite}(\Uparrow, A) = \text{self}[f^A_{acc}]/(\Uparrow[\neg f^A_{acc}])^*/\text{self}[f^A_{acc}]$.

Rewrite the query inductively.

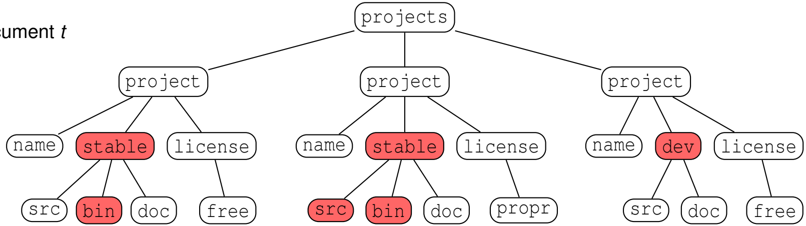
$\square$

Rewriting Queries

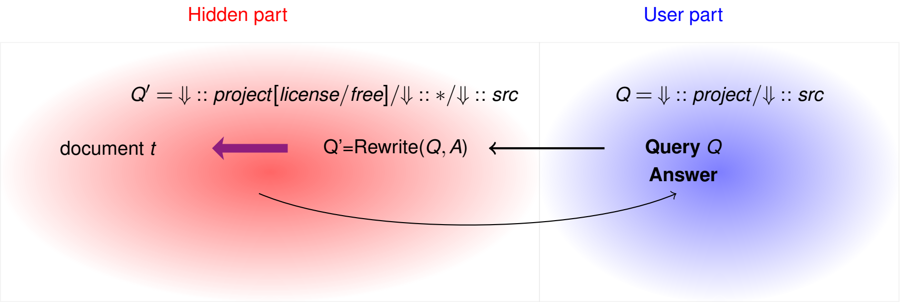
Hidden part | User part

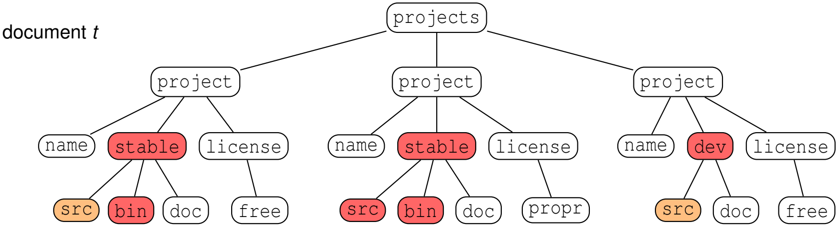


document *t*



Rewriting Queries





Outline

1. Non-materialized views and query rewriting

2. Comparing Access Control Policies

Comparing access control policies

Definition

Two annotations A_1 and A_2 over DTD D are *equivalent* iff they hide the same nodes:

$$A_1 \equiv^D A_2 \text{ iff } \forall t \in L(D).\ A_1(t) = A_2(t)$$

Comparing access control policies

Definition

Two annotations A_1 and A_2 over DTD D are *equivalent* iff they hide the same nodes:

$$A_1 \equiv^D A_2 \text{ iff } \forall t \in L(D).\ A_1(t) = A_2(t)$$

Proposition

Testing equivalence of annotations is *EXPTIME*-complete.

Proof.

This problem is polynomially equivalent to the problem of equivalence of $\mathcal{X}$*Reg* filters over a DTD. □

Comparing Access control policies

Definition

A_1 and A_2 annotations over DTD D. A_1 is 1-*restriction* of A_2 in the presence of D, denoted

$$A_1 \preccurlyeq_1^D A_2 \text{ iff } \forall t \in L(D).\ N_{A_1(t)} \subseteq N_{A_2(t)}$$

Intuition:

The simplest way for comparing two annotations:
A_1 is more "restrictive" than A_2 if it shows no element hidden by A_2.

Proposition

Testing 1-restriction is *EXPTIME*-complete.

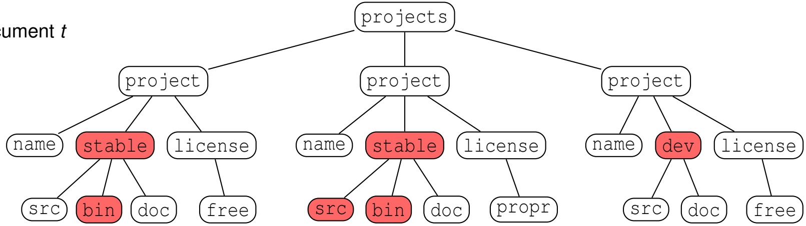
Does $\preccurlyeq_1$ ensure the properties we expect?

Example

$\texttt{projects} \to \texttt{project}^*$
$\texttt{project} \to \texttt{name}, (\texttt{stable} \mid \texttt{dev}), \texttt{license}$
$\quad A_0(\texttt{project}, \texttt{stable}) = \textsf{false}$
$\quad A_0(\texttt{project}, \texttt{dev}) = \textsf{false}$
$\texttt{license} \to \texttt{free} \mid \texttt{propr}$

$\texttt{stable} \to \texttt{src}, \texttt{bin}, \texttt{doc}$
$\quad A_0(\texttt{stable}, \texttt{src}) = [\Uparrow^*::\texttt{project}/\Downarrow^*::\texttt{free}]$
$\quad A_0(\texttt{stable}, \texttt{doc}) = \textsf{true}$
$\texttt{dev} \to \texttt{src}, \texttt{doc}$
$\quad A_0(\texttt{dev}, \texttt{src}) = [\Uparrow^*::\texttt{project}/\Downarrow^*::\texttt{free}]$
$\quad A_0(\texttt{dev}, \texttt{doc}) = \textsf{true}$

document t



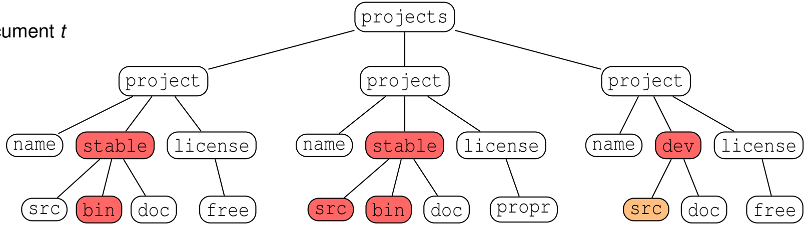
Does $\preccurlyeq_1$ ensure the properties we expect?

Example

$\texttt{projects} \rightarrow \texttt{project}^*$
$\texttt{project} \rightarrow \texttt{name}, (\texttt{stable} \mid \texttt{dev}), \texttt{license}$
$\quad A_0(\texttt{project}, \texttt{stable}) = \text{false}$
$\quad A_0(\texttt{project}, \texttt{dev}) = \text{false}$
$\texttt{license} \rightarrow \texttt{free} \mid \texttt{propr}$

$\texttt{stable} \rightarrow \texttt{src}, \texttt{bin}, \texttt{doc}$
$\quad A_0(\texttt{stable}, \texttt{src}) = [\Uparrow^*{::}\texttt{project}/\Downarrow^*{::}\texttt{free}]$
$\quad A_0(\texttt{stable}, \texttt{doc}) = \text{true}$
$\texttt{dev} \rightarrow \texttt{src}, \texttt{doc}$
$\quad A_0(\texttt{dev}, \texttt{src}) = \cancel{[\Uparrow^*{::}\texttt{project}/\Downarrow^*{::}\texttt{free}]}\ \text{false}$
$\quad A_0(\texttt{dev}, \texttt{doc}) = \text{true}$

document t



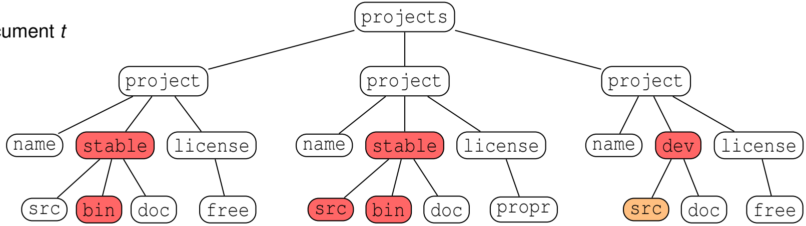
Does $\preccurlyeq_1$ ensure the properties we expect?

Example

$\mathtt{projects} \rightarrow \mathtt{project}^*$
$\mathtt{project} \rightarrow \mathtt{name}, (\mathtt{stable} \mid \mathtt{dev}), \mathtt{license}$
$\quad A_0(\mathtt{project}, \mathtt{stable}) = \mathsf{false}$
$\quad A_0(\mathtt{project}, \mathtt{dev}) = \mathsf{false}$
$\mathtt{license} \rightarrow \mathtt{free} \mid \mathtt{propr}$

$\mathtt{stable} \rightarrow \mathtt{src}, \mathtt{bin}, \mathtt{doc}$
$\quad A_0(\mathtt{stable}, \mathtt{src}) = [\Uparrow^*::\mathtt{project}/\Downarrow^*::\mathtt{free}]$
$\quad A_0(\mathtt{stable}, \mathtt{doc}) = \mathsf{true}$
$\mathtt{dev} \rightarrow \mathtt{src}, \mathtt{doc}$
$\quad A_0(\mathtt{dev}, \mathtt{src}) = \cancel{[\Uparrow^*::\mathtt{project}/\Downarrow^*::\mathtt{free}]}$ false
$\quad A_0(\mathtt{dev}, \mathtt{doc}) = \mathsf{true}$

document t



$\Rightarrow$User can select all projects under free license that are not stable !

An information-oriented comparison

Argument

A_1 should be more "restrictive" than A_2 if every information inferred from A_1 can be inferred from A_2.

Definition

A_1 and A_2 annotations over DTD D. A_1 is 2-*restriction* of A_2 in the presence of D, denoted

$$A_1 \preccurlyeq_2^D A_2 \text{ iff } \forall Q_1 \exists Q_2.\ \forall t \in L(D).\ Ans(Q_1, A_1(t)) = Ans(Q_2, A_2(t))$$

An information-oriented comparison

Definition

A_1 and A_2 annotations over DTD D. A_1 is *2-restriction* of A_2 in the presence of D, denoted

$$A_1 \preccurlyeq_2^D A_2 \text{ iff } \forall Q_1 \exists Q_2.\ \forall t \in L(D).\ Ans(Q_1, A_1(t)) = Ans(Q_2, A_2(t))$$

Theorem

This property is undecidable.

An information-oriented comparison

Definition

A_1 and A_2 annotations over DTD D. A_1 is *2-restriction* of A_2 in the presence of D, denoted

$$A_1 \preccurlyeq_2^D A_2 \text{ iff } \forall Q_1 \exists Q_2.\ \ \forall t \in L(D).\ Ans(Q_1, A_1(t)) = Ans(Q_2, A_2(t))$$

Theorem

This property is undecidable.

Alternative characterization

$A_1 \preccurlyeq_2^D A_2$ if and only if
$\exists f. \forall t \models D \forall n \in N_{A_2}(t).(n, A_2(t)) \models f \iff n \in N_{A_1}(t)$

$\Longrightarrow$: if filter f is provided, then one can verify the property in *EXPTIME*

An information-oriented comparison

Definition

A_1 and A_2 annotations over DTD D. A_1 is *2-restriction* of A_2 in the presence of D, denoted

$$A_1 \preccurlyeq_2^D A_2 \text{ iff } \forall Q_1 \exists Q_2.\ \forall t \in L(D).\ Ans(Q_1, A_1(t)) = Ans(Q_2, A_2(t))$$

Theorem

This property is undecidable.

Theorem

However, for non-recursive DTDs, 2-restriction can be tested in *EXPTIME*

Further work

- implementation
- update propagation
- richer schema and query language
- other view formalisms