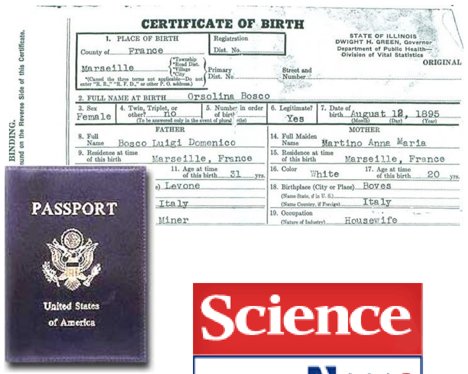
Querying and storing XML

Week 8
Provenance
March 12-15, 2013

What is provenance?

- Evidence of
  - Origin
  - History
  - Authenticity
  - Integrity
  - Value



Why is provenance important for *data*?

- For traditional (paper) information:
  - Creation process leaves “paper trail”
  - Easier to detect modification, copying, forgery
  - Can *usually* judge a book by its cover
- For electronic information:
  - Often no such thing as a “bit trail”
  - Easy to forge, plagiarize, alter data undetected
  - Can't judge a database by its cover - **there isn't one**
- Provenance essential for judging quality of data

Provenance failures can be expensive


UAL Shares Fall as Old Story Surfaces Online - WSJ.com - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://online.wsj.com/article/SB12

SEPTEMBER 9, 2008

UAL Shares Fall as Old Story Surfaces Online

The mysterious appearance on the Internet of a nearly six-year-old news story about UAL Corp.'s 2002 bankruptcy-court filing caused investors to dump the stock Monday.

After trading near $12.50 a share early Monday, stock in United Airlines' parent quickly fell to $3 on the Nasdaq Stock Market on heavy volume before trading was halted and the company issued a statement saying that reports of a new Chapter 11 filing were "completely untrue."

Once trading resumed 90 minutes later, UAL shares rebounded, but they still closed off 11% for the day at $10.92. Nasdaq, a unit of Nasdaq OMX Group Inc., ...

Especially important for scientific data

SCIENTIFIC PUBLISHING

A Scientist's Nightmare: Software Problem Leads to Five Retractions

Until recently, Geoffrey Chang's career was on a trajectory most young scientists only dream about. In 1999, at the age of 28, the protein crystallographer landed a faculty position at the prestigious Scripps Research Institute in San Diego, California. The next year, in a ceremony at the White House, Chang received a 2001 *Science* paper, which described the structure of a protein called MsbA, isolated from the bacterium *Escherichia coli*. MsbA belongs to a huge and ancient family of molecules that use energy from adenosine triphosphate to transport molecules across cell membranes. These so-called ABC transporters perform many

Provenance in Databases

- Provenance models extensively studied in *relational databases*
  - Why-provenance
  - Where-provenance
  - How-provenance
  - ....?
- Will examine provenance models for relational queries first
  - following recent survey [Cheney, Chiticariu, Tan 2009]

Why-provenance

(Buneman, Khanna, Tan 2001)

- *Why-provenance*: shows input data *witnessing* existence of output data

R

A	B	C
1	2	2
1	2	3
2	3	4

S

C	D
1	2
2	2
2	3

R JOIN S

A	B	C	D
1	2	2	2
1	2	2	3

Why-provenance

(Buneman, Khanna, Tan 2001)

- *Why-provenance*: shows input data *witnessing* existence of output data
- = subset of input that is "enough" to generate output

R

A	B	C
1	2	2
1	2	3
2	3	4

S

C	D
1	2
2	2
2	3

R JOIN S

A	B	C	D
1	2	2	2
1	2	2	3

Why-provenance

(Buneman, Khanna, Tan 2001)

- *Why-provenance*: shows input data *witnessing* existence of output data
- = subset of input that is "enough" to generate output

R

A	B	C
1	2	2
1	2	3
2	3	4

S

C	D
1	2
2	2
2	3

R JOIN S

A	B	C	D
1	2	2	2
1	2	2	3

Where-provenance

(Buneman, Khanna, Tan 2001)

- *Where-provenance*: tracks where data in output *comes from*

R

A	B	C
1	2	2
1	2	3
2	3	4

S

C	D
1	2
2	2
2	3

R JOIN S

A	B	C	D
1	2	2	2
1	2	2	3

Where-provenance

(Buneman, Khanna, Tan 2001)

- *Where-provenance*: tracks where data in output *comes from*

R

A	B	C
1	2	2
1	2	3
2	3	4

S

C	D
1	2
2	2
2	3

R JOIN S

A	B	C	D
1	2	2	2
1	2	2	3

Where-provenance

(Buneman, Khanna, Tan 2001)

- Can think of provenance as "links"

R

A	B	C
1	2	2
1	2	3
2	3	4

S

C	D
1	2
2	2
2	3

R JOIN S

A	B	C	D
1	2	2	2
1	2	2	3

Where-provenance

(Buneman, Khanna, Tan 2001)

- Can think of provenance as "links"
- or propagated "annotations"

R

A	B	C
1	2	2
1	2	3
2	3	4

S

C	D
1	2
2	2
2	3

R JOIN S

A	B	C	D
1	2	2	2
1	2	2	3

Where-provenance

(Buneman, Khanna, Tan 2001)

- Not invariant under query equivalence

R

A	B	C
1	2	2
1	2	3
2	3	4

S

C	D
1	2
2	2
2	3

```
SELECT r.A,r.B,r.C,s.D
FROM R r, S s
WHEERE r.C = s.C
```

A	B	C	D
1	2	2	2
1	2	2	3

Where-provenance

(Buneman, Khanna, Tan 2001)

- Not invariant under query equivalence

R

A	B	C
1	2	2
1	2	3
2	3	4

S

C	D
1	2
2	2
2	3

```
SELECT r.A,r.B,s.C,s.D
FROM R r, S s
WHEERE r.C = s.C
```

A	B	C	D
1	2	2	2
1	2	2	3

Early work

- Definitions were very complicated

Definition 4.1 (Tuple Derivation for an Operator). Let Op be any relational operator over tables $T_1, \ldots, T_m$, and let $T = Op(T_1, \ldots, T_m)$ be the table that results from applying Op to $T_1, \ldots, T_m$. Given a tuple $t \in T$, we define *t's derivation in* $T_1, \ldots, T_m$ *according to* Op to be $Op^{-1}_{\langle T_1, \ldots T_m \rangle}(t) = \langle T^*_1, \ldots T^*_m \rangle$, where $T^*_1, \ldots, T^*_m$ are *maximal* subsets of $T_1, \ldots, T_m$ such that

(a) $Op(T^*_1, \ldots T^*_m) = \{t\}$.

(b) $\forall T^*_i : \forall t^* \in T^*_i : Op(T^*_1, \ldots, \{t^*\}, \ldots, T^*_m) \neq \emptyset$.

We also say that $Op^{-1}_{T_i}(t) = T^*_i$ is *t's derivation in* T_i, and each tuple t^* in T^*_i *contributes to* t, for $i = 1..m$.

Early work

- Definitions were very complicated.

Definition 6. (Witness Basis) Consider a normal form query Q. The *witness basis* for a singular value t with respect to Q and D, denoted as $W_{Q,D}(t)$, is:

(1) If Q is of the form $Q_1 \sqcup ... \sqcup Q_n$ then $W_{Q,D}(t) = W_{Q_1,D}(t) \cup ... \cup W_{Q_n,D}(t)$.

(2) If Q is of the form $\{e \mid p_0 \in e_0, ..., p_n \in e_n, \mathit{condition}\}$, let Ψ be the set of all valuations on the variables of Q such that "where" clause of Q holds under each valuation in Ψ. Then, $W_{Q,D}(t) = \{[\![p_0]\!]_\psi \sqcup ... \sqcup [\![p_n]\!]_\psi \mid \psi \in \Psi, t = [\![e]\!]_\psi\}$. Note that e_i $(0 \leq i \leq n)$ is a database constant since Q is in normal form.

(3) Otherwise, $W_{Q,D}(t) = \{\}$.

More generally, for any well-formed query Q, we can define the witness basis by extending (2) as follows. We partition the set of $p_i \in e_i$ in the "where" clause of Q into two parts: $S_1 = \{p_i \mid e_i \text{ is the database constant } D\}$ and $S_2 = \{(p_i, e_i) \mid p_i \text{ is a pattern matched against a query } e_i\}$. We use $p_0^1, ..., p_k^1$ to denote the members of S_1 and $(p_0^2, e_0^2), ..., (p_m^2, e_m^2)$ to denote the members of S_2. Let Ψ be the set of all valuations on the variables of Q such that for each valuation in Ψ, "where" clause of Q holds. Then $W_{Q,D}(t) = \{P_1 \sqcup P_2 \mid \psi \in \Psi, t \sqsubseteq [\![e]\!]_\psi, P_1 = [\![p_0^1]\!]_\psi \sqcup ... \sqcup [\![p_k^1]\!]_\psi, P_2 = w_1 \sqcup ... \sqcup w_m \text{ where } w_i \in W_{\psi(e_i^2),D}([\![p_i^2]\!]_\psi)\}$. For a compound value t, the witness basis is the product of individual witness basis of singular values making up t. That is, consider $t = t_1 \sqcup ... \sqcup t_m$ where each t_i is singular. Then $W_{Q,D}(t) = \{w_1 \sqcup ... \sqcup w_m \mid w_i \in W_{Q,D}(t_i)\}$. □

Early work

- Definitions were very complicated.

Definition 8. (Derivation Basis) Consider a normal form query Q. The *derivation basis* for $l{:}v$ where v is an atomic value, denoted as $\Gamma_{Q,D}(l : v)$ with respect to Q and D, is defined as below:

(1) If $Q = Q_1 \sqcup ... \sqcup Q_n$ then $\Gamma_{Q,D}(l : v) = \Gamma_{Q_1,D}(l : v) \cup ... \cup \Gamma_{Q_n,D}(l : v)$.

(2) If Q has the form $\{e \mid p_0 \in e_0, ..., p_n \in e_n, \mathit{condition}\}$, let Ψ be the set of valuations on the variables of Q such that the "where" clause of Q holds under each valuation and $\psi(e)$ contains $l{:}v$. For each $\psi \in \Psi$, let p_{x_ψ} denote the path in e that points to a variable x_ψ such that there exists p' and p'' so that $l = p'.p''$ and $\psi(p_{x_\psi}) = p'$ and $\psi(x_\psi)(p'') = v$. Then, $\Gamma_{Q,D}(l : v) = \{([\![p_0]\!]_\psi \sqcup ... \sqcup [\![p_n]\!]_\psi, S) \mid \psi \in \Psi, S = \{\psi(p_i').p'' \mid p_i' \text{ is the path that points to variable } x_\psi \text{ in pattern } p_i, 0 \leq i \leq n\}\}$.

(3) Otherwise, $\Gamma_{Q,D}(l : v) = \{\}$.

More generally, the derivation basis of $l{:}v$ where v is a compound value is defined to be the derivation basis of all possible (path,value) pairs $p'{:}v'$ such that $p'{:}v'$ points to a value in v. The derivation basis for multiple (path,value) pairs is defined to be the product of the derivation basis of individual (path,value) pairs. That is, $\Gamma_{Q,D}(p_1{:}v_1, p_2{:}v_2) = \Gamma_{Q,D}(p_1{:}v_1) * \Gamma_{Q,D}(p_2{:}v_2) = \{(w_1 \sqcup w_2, P_1 \cup P_2) \mid (w_1, P_1) \in \Gamma_{Q,D}(p_1{:}v_1), (w_2, P_2) \in \Gamma_{Q,D}(p_2{:}v_2)\}$. □

Ordinary relational algebra

$$
\begin{aligned}
(\{t\})(I) &= \{t\} \\
R(I) &= I(R) \\
(\sigma_\theta(Q))(I) &= \{t \in Q(I) \mid \theta(t)\} \\
(\pi_U(Q))(I) &= \{t[U] \mid t \in Q(I)\} \\
(Q_1 \bowtie Q_2)(I) &= \{t \mid t[U_1] \in Q_1(I), t[U_2] \in Q_2(I)\} \\
(Q_1 \cup Q_2)(I) &= Q_1(I) \cup Q_2(I) \\
(\rho_{A \mapsto B}(Q))(I) &= \{t[A \mapsto B] \mid t \in Q(I)\}
\end{aligned}
$$

Datalog

- Queries can also be written in a logical form called *Datalog* (subset of Prolog)
- $A(x_1,...,x_n)$:- $R(y_1,...,y_m)$, ..., $S(z_1,...,z_k)$
  - (subject to some restrictions...)
- **Theorem**: Relational algebra, relational calculus and nonrecursive Datalog are equally expressive

Example

- Two (equivalent) queries on a small table

Instance I:

R

	A	B
t:	1	2
t':	1	3
t'':	4	2

Two equivalent queries:

$Q : Ans(x,y) :- R(x,y).$

$Q' : Ans(x,y) :- R(x,y), R(x,z).$

Output of $Q(I)$, $Q'(I)$:

A	B
1	2
1	3
4	2

Why-provenance
[Buneman et al. 2001]

- Propagate sets of *witnesses*
  - elements of {J ⊆ I | t ∈ Q(J)}

$$\mathsf{Why}(\{t\}, I, \{u\}) = \begin{cases} \{\emptyset\}, & \text{if } (t = u), \\ \emptyset, & \text{otherwise.} \end{cases}$$

$$\mathsf{Why}(R, I, t) = \begin{cases} \{\{(R,t)\}\}, & \text{if } (t \in R(I)), \\ \emptyset, & \text{otherwise.} \end{cases}$$

$$\mathsf{Why}(\sigma_\theta(Q), I, t) = \begin{cases} \mathsf{Why}(Q, I, t), & \text{if } \theta(t), \\ \emptyset, & \text{otherwise.} \end{cases}$$

$$\mathsf{Why}(\pi_U(Q), I, t) = \bigcup \{\mathsf{Why}(Q, I, u) \mid u \in Q(I), t = u[U]\}$$

$$\mathsf{Why}(\rho_{A \mapsto B}(Q), I, t) = \mathsf{Why}(Q, I, t[B \mapsto A])$$

$$\mathsf{Why}(Q_1 \bowtie Q_2, I, t) = \mathsf{Why}(Q_1, I, t[U_1]) \uplus \mathsf{Why}(Q_2, I, t[U_2])$$

$$\mathsf{Why}(Q_1 \cup Q_2, I, t) = \mathsf{Why}(Q_1, I, t) \cup \mathsf{Why}(Q_2, I, t))$$

Why-provenance
[Buneman et al. 2001]

- Propagate sets of *witnesses*
  - elements of {J ⊆ I | t ∈ Q(J)}

$\mathsf{Why}(\{t\}, I, \{u\}) = \{\{\emptyset\},$ if $(t = u)$, ... otherwise.

Pairwise union of sets of justifications
S ⋓ T = {J ∪ K | J ∈ S, K ∈ T}

if $(t \in R(I))$, otherwise.

if $\theta(t)$, otherwise.

$u) \mid u \in Q(I), t = u[U]\}$

$$\mathsf{Why}(\rho_{A \mapsto B}(Q), I, t) = \mathsf{Why}(Q, I, t[B \mapsto A])$$

$$\mathsf{Why}(Q_1 \bowtie Q_2, I, t) = \mathsf{Why}(Q_1, I, t[U_1]) \uplus \mathsf{Why}(Q_2, I, t[U_2])$$

$$\mathsf{Why}(Q_1 \cup Q_2, I, t) = \mathsf{Why}(Q_1, I, t) \cup \mathsf{Why}(Q_2, I, t))$$

Why-provenance

- Also sensitive to query rewriting

Instance I:

R

	A	B
t:	1	2
t':	1	3
t'':	4	2

Two equivalent queries:

$Q : Ans(x,y) :- R(x,y).$

$Q' : Ans(x,y) :- R(x,y), R(x,z).$

Output of $Q(I)$, $Q'(I)$:

A	B
1	2
1	3
4	2

Instance I:

R

	A	B
t:	1	2
t':	1	3
t'':	4	2

Output of $Q(I)$

A	B	*why*
1	2	$\{\{t\}\}$
1	3	$\{\{t'\}\}$
4	2	$\{\{t''\}\}$

Output of $Q'(I)$

A	B	*why*
1	2	$\{\{t\}, \{t, t'\}\}$
1	3	$\{\{t'\}, \{t, t'\}\}$
4	2	$\{\{t''\}\}$

Why-provenance

- Also sensitive to query rewriting

Instance I:

R

	A	B
t:	1	2
t':	1	3
t'':	4	2

Two e[illegible]

$Q : A$[illegible]

$Q' :$[illegible]

Output of $Q(I)$, $Q'(I)$:

A	B
[illegible]	[illegible]

Can recover by removing non-minimal witnesses

Instance I:

R

	A	B
t:	1	2
t':	1	3
t'':	4	2

$Q(I)$

A	B	*why*
1	2	$\{\{t\}\}$
1	3	$\{\{t'\}\}$
4	2	$\{\{t''\}\}$

$Q'(I)$

A	B	*why*
1	2	$\{\{t\},\{t,t'\}\}$
1	3	$\{\{t'\},\{t,t'\}\}$
4	2	$\{\{t''\}\}$

Where-provenance

[Buneman et al. 2001]

- Propagate field-level annotation sets

$$\begin{aligned}
\mathsf{Where}(\{u\},I,t) &= \begin{cases}(A:\emptyset)_{A\in U}, & \text{if } t=u\\ \bot, & \text{otherwise}\end{cases}\\
\mathsf{Where}(R,I,t) &= \begin{cases}(A:\{(R,t,A)\})_{A\in U}, & \text{if } t\in I(R)\\ \bot, & \text{otherwise}\end{cases}\\
\mathsf{Where}(\sigma_\theta(Q),I,t) &= \begin{cases}\mathsf{Where}(Q,I,t), & \text{if } \theta(t)\\ \bot, & \text{otherwise}\end{cases}\\
\mathsf{Where}(\pi_U(Q),I,t) &= \bigsqcup_L\{\mathsf{Where}(Q,I,u)[U] \mid u[U]=t\}\\
\mathsf{Where}(\rho_{B\mapsto C}(Q),I,t) &= (A:\mathsf{Where}(Q,I,t[C\mapsto B])\cdot(A[C\mapsto B]))_{A\in U}\\
\mathsf{Where}(Q_1\bowtie Q_2,I,t) &= \mathsf{Where}(Q_1,I,t[U_1])\sqcup_S \mathsf{Where}(Q_2,I,t[U_2])\\
\mathsf{Where}(Q_1\cup Q_2,I,t) &= \mathsf{Where}(Q_1,I,t)\sqcup_L \mathsf{Where}(Q_2,I,t)
\end{aligned}$$

Where-provenance

- May not be preserved by query equivalence

Instance I:

R

	A	B
t:	1	2
t':	1	3
t'':	4	2

Two equivalent queries:

$Q : Ans(x,y) :- R(x,y).$

$Q' : Ans(x,y) :- R(x,y), R(x,z).$

Output of $Q(I)$, $Q'(I)$:

A	B
1	2
1	3
4	2

Annotated instance I^a:

R

	A	B
t:	1^{a_1}	2^{a_2}
t':	1^{a_3}	3^{a_4}
t'':	4^{a_5}	2^{a_6}

Output of $Q(I^a)$

A	B
1^{a_1}	2^{a_2}
1^{a_3}	3^{a_4}
4^{a_5}	2^{a_6}

Output of $Q'(I^a)$

A	B
$1^{a_1,a_3}$	2^{a_2}
$1^{a_1,a_3}$	3^{a_4}
4^{a_5}	2^{a_6}

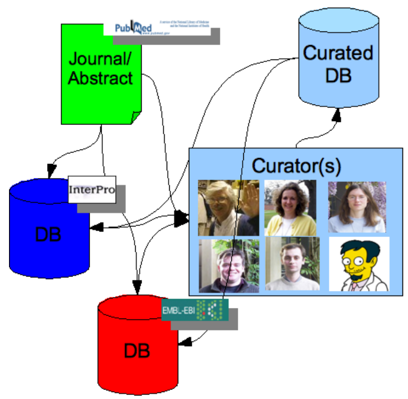
Provenance and XML

- Early work on provenance (why/where) focused on determinstic *semistructured model*
  - Similar to (special case of) XML
- Advantages:
  - XML more general; nodes easily addressed
- Complications:
  - Little work on prov for XPath/XQuery, or other XML standards
- Next topic: provenance for **updated data**

Provenance for curated data

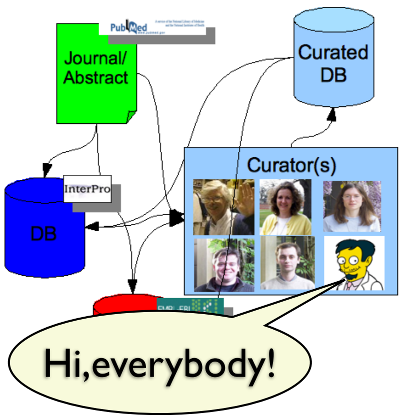
Curated databases

- Many bio-medical databases are **curated**
  - data entered, checked manually
  - high-quality
  - but **expensive**
  - provenance, versioning important
  - lots of (re)implementation effort



Curated databases

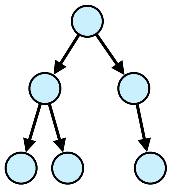
- Many bio-medical databases are **curated**
  - data entered, checked manually
  - high-quality
  - but **expensive**
  - provenance, versioning important
  - lots of (re)implementation effort



Provenance

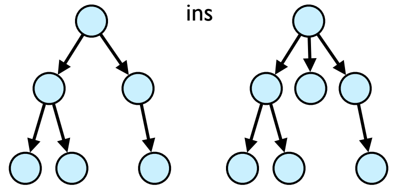
- Idea: Instead of trying to allow only "good" contributors
  - allow anyone to contribute
  - but record what they did
- Allows "auditing" after-the-fact
  - can discard or approve changes
- May combine with access control
  - allow retrospective analysis of trusted contributors

Copy-paste provenance



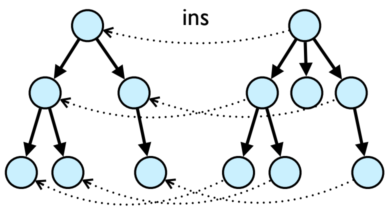
- As data (tree) is updated, record "links" identifying "same" data in consecutive versions

Copy-paste provenance



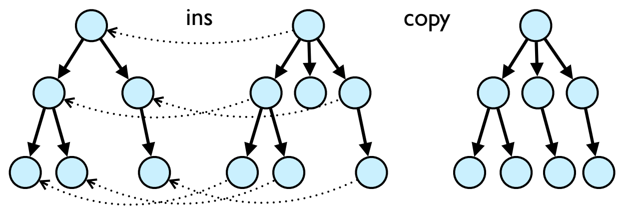
- As data (tree) is updated, record "links" identifying "same" data in consecutive versions

Copy-paste provenance



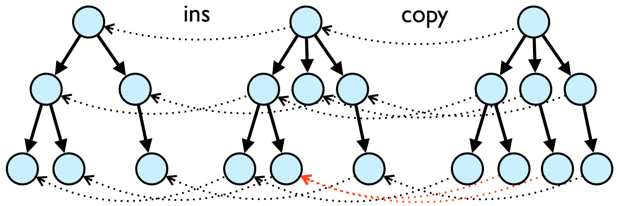
- As data (tree) is updated, record "links" identifying "same" data in consecutive versions

Copy-paste provenance



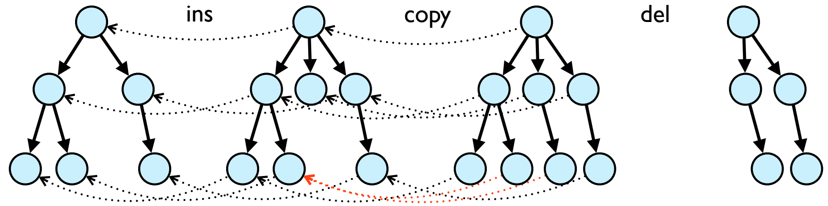
- As data (tree) is updated, record "links" identifying "same" data in consecutive versions

Copy-paste provenance



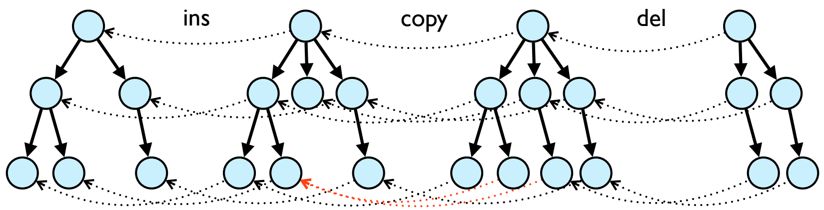
- As data (tree) is updated, record "links" identifying "same" data in consecutive versions

Copy-paste provenance



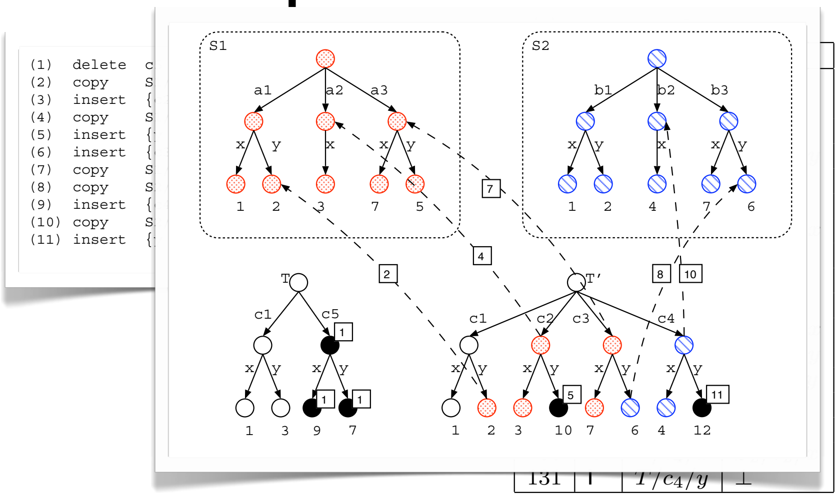
- As data (tree) is updated, record "links" identifying "same" data in consecutive versions

Copy-paste provenance



- As data (tree) is updated, record "links" identifying "same" data in consecutive versions

Relational representation



Relational representation

```
(1)  delete  c5              from T;
(2)  copy    S1/a1/y         into T/c1/y;
(3)  insert  {c2 : {}}       into T;
(4)  copy    S1/a2           into T/c2;
(5)  insert  {y : 10}        into T/c2;
(6)  insert  {c3 : {}}       into T;
(7)  copy    S1/a3           into T/c3;
(8)  copy    S2/b3/y         into T/c3/y;
(9)  insert  {c4 : {}}       into T;
(10) copy    S2/b2           into T/c4;
(11) insert  {y : 12}        into T/c4;
```

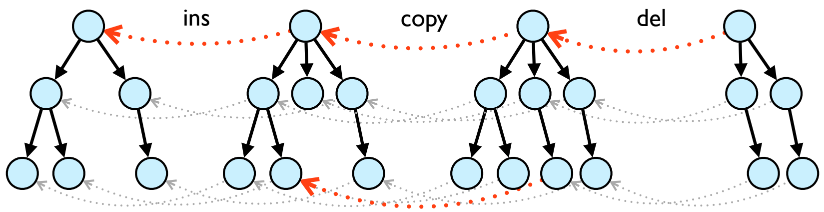
(a) Prov

Tid	*Op*	*Loc*	*Src*
121	D	T/c_5	$\bot$
121	D	$T/c_5/x$	$\bot$
121	D	$T/c_5/y$	$\bot$
122	C	$T/c_1/y$	$S_1/a_1/y$
123	I	T/c_2	$\bot$
124	C	T/c_2	S_1/a_2
124	C	$T/c_2/x$	$S_1/a_2/x$
125	I	$T/c_2/y$	$\bot$
126	I	T/c_3	$\bot$
127	C	T/c_3	S_1/a_3
127	C	$T/c_3/x$	$S_1/a_3/x$
127	C	$T/c_3/y$	$S_1/a_3/y$
128	C	$T/c_3/y$	$S_2/b_3/y$
129	I	T/c_4	$\bot$
130	C	T/c_4	S_2/b_2
130	C	$T/c_4/x$	$S_2/b_2/x$
131	I	$T/c_4/y$	$\bot$

Performance

- Isn't this expensive?
  - storing one edge per copied node
- Two optimizations:
  - *Hierarchical* provenance: inheriting inferrable annotations
  - *Transactional* provenance: storing only "diff" between "committed" versions, not intermediate steps

Hierarchical provenance



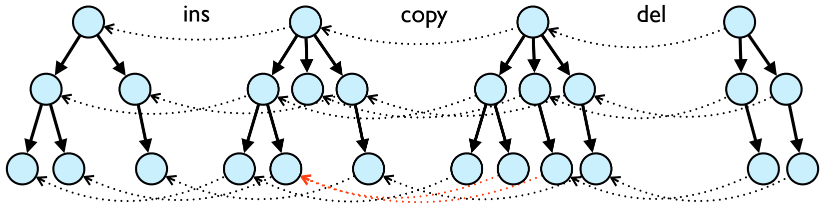
- Infer that prov of child is child of prov
- Only store **important** (non-inferrable) edges

Hierarchical provenance

$$\mathsf{Infer}(t,p) \leftarrow \neg(\exists x,q.\mathsf{HProv}(t,x,p,q))$$
$$\mathsf{Prov}(t,op,p,q) \leftarrow \mathsf{HProv}(t,op,p,q).$$
$$\mathsf{Prov}(t,\mathsf{C},p/a,q/a) \leftarrow \mathsf{Prov}(t,\mathsf{C},p,q),\mathsf{Infer}(t,p).$$
$$\mathsf{Prov}(t,\mathsf{I},p/a,\bot) \leftarrow \mathsf{Prov}(t,\mathsf{I},p,\bot),\mathsf{Infer}(t,p).$$
$$\mathsf{Prov}(t,\mathsf{D},p/a,\bot) \leftarrow \mathsf{Prov}(t,\mathsf{D},p,\bot),\mathsf{Infer}(t,p).$$

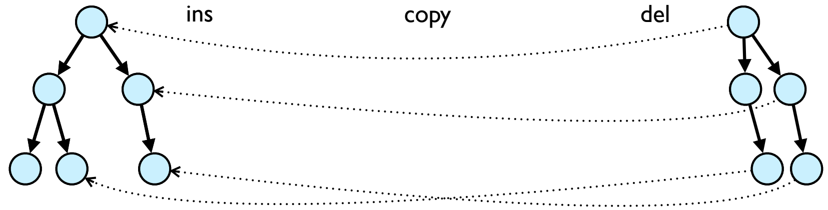
- Infer that prov of child is child of prov
- Only store **important** (non-inferrable) edges

Transactional provenance



- Require users to commit "checkpoints" (official versions)
- Concatenate edges between versions

Transactional provenance



- Require users to commit "checkpoints" (official versions)
- Concatenate edges between versions

Effect of optimizations

(b) Prov

Tid	Op	Loc	Src
121	D	T/c_5	$\bot$
121	D	$T/c_5/x$	$\bot$
121	D	$T/c_5/y$	$\bot$
121	C	$T/c_1/y$	$S_1/a_1/y$
121	C	T/c_2	S_1/a_2
121	C	$T/c_2/x$	$S_1/a_2/x$
121	I	$T/c_2/y$	$\bot$
121	C	T/c_3	S_1/a_3
121	C	$T/c_3/x$	$S_1/a_3/x$
121	C	$T/c_3/y$	$S_2/b_3/y$
121	C	T/c_4	S_2/b_2
121	C	$T/c_4/x$	$S_2/b_2/x$
121	I	$T/c_4/y$	$\bot$

Transactional

(c) HProv

Tid	Op	Loc	Src
121	D	T/c_5	$\bot$
122	C	$T/c_1/y$	$S_1/a_1/y$
123	I	T/c_2	$\bot$
124	C	T/c_2	S_1/a_2
125	I	$T/c_2/y$	$\bot$
126	I	T/c_3	$\bot$
127	C	T/c_3	S_1/a_3
128	C	$T/c_3/y$	$S_2/b_3/y$
129	I	T/c_4	$\bot$
130	C	T/c_4	S_2/b_2
131	I	$T/c_4/y$	$\bot$

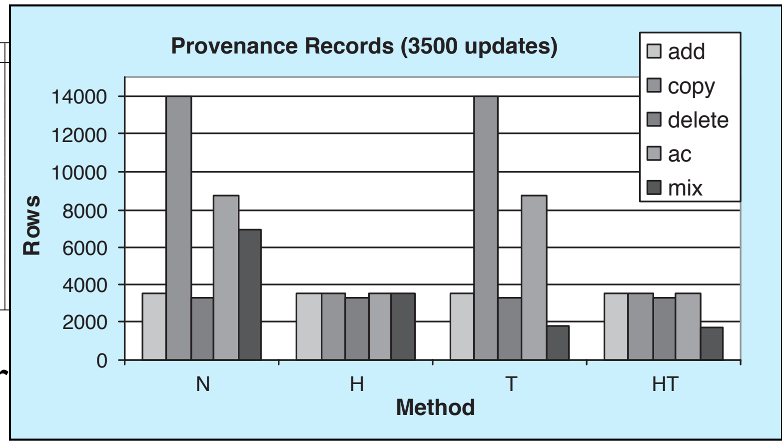
Hierarchical

(d) HProv

Tid	Op	Loc	Src
121	D	T/c_5	$\bot$
121	C	$T/c_1/y$	$S_1/a_1/y$
121	C	T/c_2	S_1/a_2
121	I	$T/c_2/y$	$\bot$
121	C	T/c_3	S_1/a_3
121	C	$T/c_3/y$	$S_2/b_3/y$
121	C	T/c_4	S_2/b_2
121	I	$T/c_4/y$	$\bot$

Both

Effect of optimizations



Queries

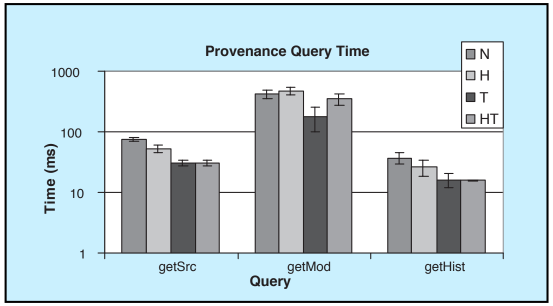
$$
\begin{array}{lll}
\mathsf{Unch}(t,p) & \leftarrow & \neg(\exists x,q.\mathsf{Prov}(t,x,p,q)). \\
\mathsf{Ins}(t,p) & \leftarrow & \mathsf{Prov}(t,\mathsf{I},p,\bot) \\
\mathsf{Del}(t,p) & \leftarrow & \mathsf{Prov}(t,\mathsf{D},p,\bot) \\
\mathsf{Copy}(t,p,q) & \leftarrow & \mathsf{Prov}(t,\mathsf{C},p,q)
\end{array}
$$

$$
\begin{array}{lll}
\mathsf{Trace}(p,t,p,t). & & \\
\mathsf{Trace}(p,t,q,u) & \leftarrow & \mathsf{Trace}(p,t,r,s), \mathsf{Trace}(r,s,q,u). \\
\mathsf{Trace}(p,t,q,t-1) & \leftarrow & \mathsf{From}(t,p,q).
\end{array}
$$

$$
\begin{array}{lll}
\mathsf{Src}(p) & = & \{u \mid \exists q.\mathsf{Trace}(p,t_{now},q,u), \mathsf{Ins}(u,q)\} \\
\mathsf{Hist}(p) & = & \{u \mid \exists q.\mathsf{Trace}(p,t_{now},q,u), \mathsf{Copy}(u,q)\} \\
\mathsf{Mod}(p) & = & \{u \mid \exists q.p \leq q, \mathsf{Trace}(q,t_{now},r,u), \neg\mathsf{Unch}(u,r)\}
\end{array}
$$

- Provenance queries are naturally *recursive*
  - don't know how far back into history we need to look

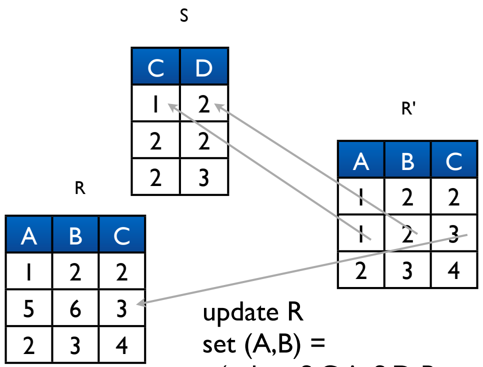
Performance



- Query performance generally *improves* with H, T, HT storage strategy
  - for H, this is somewhat surprising!
  - Cheaper to recompute inferred links than to load

Generalizing to bulk updates

[Buneman, Cheney & Vansummeren 2008]



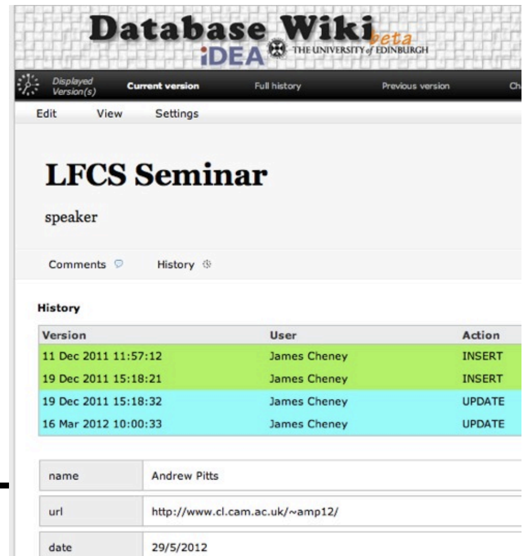

```
update R
set (A,B) =
  (select S.C A, S.D B
   from S where S.A = 1)
where R.C = 3
```

ICDT 2007/TODS 2008

Database Wiki

[Buneman, Cheney, Lindley, Müller, SIGMOD/SIGMOD Record 2011]

- Wiki-like Web application for data curation
- Archiving, copy-paste provenance "built-in"
- http://code.google.com/p/database-wiki/



Provenance & annotation for XML queries

How-provenance
(Green, Karvounakaris, Tannen 2007)

- *How-provenance:* shows how records were combined to form output

R

A	B	C	
1	2	2	a
1	2	3	b
2	3	4	c

S

C	D	
1	2	x
2	2	y
4	3	z

SELECT A,B
FROM R JOIN S

A	B	
1	2	ax+by
2	3	cz

How-provenance
(Green, Karvounakaris, Tannen 2007)

- *How-provenance:* shows how records were combined to form output

R

A	B	C	
1	2	2	a
1	2	3	b
2	3	4	c

S

C	D	
1	2	x
2	2	y
4	3	z

SELECT A,B
FROM R JOIN S

A	B	
1	2	ax+by
2	3	cz

How-provenance
(Green, Karvounakaris, Tannen 2007)

- *How-provenance:* shows how records were combined to form output

R

A	B	C	
1	2	2	a
1	2	3	b
2	3	4	c

S

C	D	
1	2	x
2	2	y
4	3	z

SELECT A,B
FROM R JOIN S

A	B	
1	2	ax+by
2	3	cz

How-provenance

(Green, Karvounakaris, Tannen 2007)

- *How-provenance:* shows how records were combined to form output

R

A	B	C	
1	2	2	a
1	2	3	b
2	3	4	c

S

C	D	
1	2	x
2	2	y
4	3	z

SELECT A,B
FROM R JOIN S

A	B	
1	2	ax+by
2	3	cz

More about how-provenance

- Formalized using *semiring-valued relations*
- Idea: Each n-tuple in relation carries an annotation from a *commutative semiring*
- *K* = (K,0,1,+,*) is a commutative semiring if:
  - (K,0,+) and (K,1,*) are commutative monoids
  - a*0 = 0 (annihilation)
  - a(b+c) = ab+ac (distributivity)

Some standard examples of semirings

- Booleans B = ({0,1},0,1,∨,∧)
- Numbers N = ({0,1,...},0,1,+,·)
- Free semiring $\mathbb{N}[X]$
  - Polynomials over X with coefficients from N
  - Formal addition, multiplication

Semiring-valued relational algebra

$$
\begin{aligned}
(\{u\})^K(I)t &= \begin{cases} 1 & t = u \\ 0 & \text{otherwise} \end{cases} \\
R^K(I)t &= I(R)(t) \\
(\sigma_\theta(Q))^K(I)t &= \theta(t) \cdot Q^K(I)t \\
(\rho_{A \mapsto B}(Q))^K(I)t &= Q^K(I)(t[B \mapsto A]) \\
(\pi_V(Q))^K(I)t &= \sum_{u \in \mathrm{supp}(Q^K(I)), u[V]=t} Q^K(I)u \\
(Q_1 \bowtie Q_2)^K(I)t &= {Q_1}^K(I)(t[U_1]) \cdot {Q_2}^K(I)(t[U_2]) \\
(Q_1 \cup Q_2)^K(I)t &= {Q_1}^K(I)t + {Q_2}^K(I)t
\end{aligned}
$$

Semiring-valued relational algebra

$$
\begin{aligned}
(\{u\})^K(I)t &= \begin{cases} 1 & t = u \\ 0 & \text{otherwise} \end{cases} \\
R^K(I)t &= I(R)(t) \\
(\sigma_\theta(Q))^K(I)t &= \theta(t) \cdot Q^K(I)t \\
(\rho_{A \mapsto B}(Q))^K(I)t &= Q^K(I \ldots \\
(\pi_V(Q))^K(I)t &= \sum_{u \in \mathrm{sup} \ldots} \\
(Q_1 \bowtie Q_2)^K(I)t &= {Q_1}^K(\ldots \\
(Q_1 \cup Q_2)^K(I)t &= {Q_1}^K(\ldots
\end{aligned}
$$

I(R) is a function from tuples t to their annotations in K

Semiring-valued relational algebra

$$
\begin{aligned}
(\{u\})^K(I)t &= \begin{cases} 1 & t = u \\ 0 & \text{otherwise} \end{cases} \\
R^K(I)t &= I(R)(t) \\
(\sigma_\theta(Q))^K(I)t &= \theta(t) \cdot Q^K(I)t \\
(\rho_{A \mapsto B}(Q))^K(I)t &= Q^K(I)(t[B \mapsto A]) \\
(\pi_V(Q))^K(I)t &= \sum_{u \in \mathrm{supp}(Q^K(I)), u[V]=t} Q^K(I)u \\
(Q_1 \bowtie Q_2)^K(I)t &= {Q_1}^K(I)(t[U_1]) \cdot {Q_2}^K(I)(t[U_2]) \\
(Q_1 \cup Q_2)^K(I)t &= {Q_1}^K(I)t + {Q_2}^K(I)t
\end{aligned}
$$

Key observation

- When K = B, we get standard set-based semantics
- When K = $\mathbb{N}$, we get standard *multiset* semantics
- When K = $\mathbb{N}$[X], we get *how-provenance* semantics

How-provenance

- Preserves multiset, but not set semantics

Instance I:

R

	A	B
t:	1	2
t':	1	3
t'':	4	2

Two equivalent queries:
$Q : Ans(x,y) :- R(x,y).$
$Q' : Ans(x,y) :- R(x,y), R(x,z).$

Output of $Q(I)$, $Q'(I)$:

A	B
1	2
1	3
4	2

Instance I:

R

	A	B
t:	1	2
t':	1	3
t'':	4	2

Output of $Q(I)$

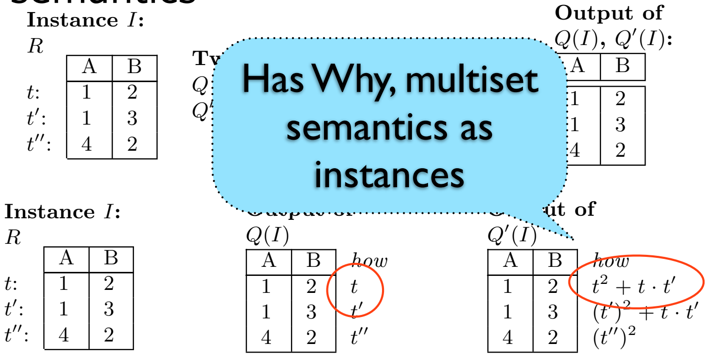
A	B	*how*
1	2	t
1	3	t'
4	2	t''

Output of $Q'(I)$

A	B	*how*
1	2	$t^2 + t \cdot t'$
1	3	$(t')^2 + t \cdot t'$
4	2	$(t'')^2$

How-provenance

- Preserves multiset, but not set semantics



Examples

- Boolean semiring

R

A	B	C	
1	2	2	T
1	2	3	T
2	3	4	T

S

C	D	
2	2	T
3	2	T
4	3	T

SELECT A,B
FROM R JOIN S

A	B	
1	2	T∧T∨T∧T
2	3	T∧T

Examples

- Boolean semiring

R

A	B	C	
1	2	2	T
1	2	3	T
2	3	4	T

S

C	D	
2	2	T
3	2	T
4	3	T

SELECT A,B
FROM R JOIN S

A	B	
1	2	T
2	3	T

Examples

- Natural numbers semiring

R

A	B	C	
1	2	2	1
1	2	3	2
2	3	4	3

S

C	D	
2	2	1
3	2	5
4	3	9

SELECT A,B
FROM R JOIN S

A	B	
1	2	1 · 1+2 · 5
2	3	3 · 9

Examples

- Natural numbers semiring

R

A	B	C	
1	2	2	1
1	2	3	2
2	3	4	3

S

C	D	
2	2	1
3	2	5
4	3	9

SELECT A,B
FROM R JOIN S

A	B	
1	2	11
2	3	27

Examples

- Polynomial semiring

R

A	B	C	
1	2	2	a
1	2	3	b
2	3	4	c

S

C	D	
2	2	x
3	2	y
4	3	z

SELECT A,B
FROM R JOIN S

A	B	
1	2	ax+by
2	3	cz

One (semi) ring to rule them all

- The polynomial semiring is "most general"
  - any other K-semantics is an instance

R

A	B	C	
1	2	2	a
1	2	3	b
2	3	4	c

S

C	D	
2	2	x
3	2	y
4	3	z

SELECT A,B
FROM R JOIN S

A	B	
1	2	ax+by
2	3	cz

One (semi) ring to rule them all

- The polynomial semiring is "most general"
  - any other K-semantics is an instance

R

A	B	C	
1	2	2	a
1	2	3	b
2	3	4	c

S

C	D	
2	2	x
3	2	y
4	3	z

SELECT A,B
FROM R JOIN S

A	B	
1	2	ax+by
2	3	cz

a=1,b=2,c=3
x=1,y=5,z=9

One (semi) ring to rule them all

- The polynomial semiring is "most general"
  - any other K-semantics is an instance

R

A	B	C	
1	2	2	1
1	2	3	2
2	3	4	3

S

C	D	
2	2	1
3	2	5
4	3	9

SELECT A,B
FROM R JOIN S

A	B	
1	2	ax+by
2	3	cz

a=1,b=2,c=3
x=1,y=5,z=9

One (semi) ring to rule them all

- The polynomial semiring is "most general"
  - any other K-semantics is an instance

R

A	B	C	
1	2	2	1
1	2	3	2
2	3	4	3

S

C	D	
2	2	1
3	2	5
4	3	9

SELECT A,B
FROM R JOIN S

A	B	
1	2	$1 \cdot 1 + 2 \cdot 5$
2	3	$3 \cdot 9$

a=1,b=2,c=3
x=1,y=5,z=9

One (semi) ring to rule them all

- The polynomial semiring is "most general"
  - any other K-semantics is an instance

R

A	B	C	
1	2	2	1
1	2	3	2
2	3	4	3

S

C	D	
2	2	1
3	2	5
4	3	9

SELECT A,B
FROM R JOIN S

A	B	
1	2	11
2	3	27

a=1,b=2,c=3
x=1,y=5,z=9

Observation

- Why-provenance can be *recovered* as an instance of how-provenance.
  - Idea: Take K = (P(P(X)), {}, {{}}, ⋓, U)

R

A	B	C	
1	2	2	{a}
1	2	3	{b}
2	3	4	{c}

S

C	D	
2	2	{x}
3	2	{y}
4	3	{z}

SELECT A,B
FROM R JOIN S

A	B	
1	2	{{a,x},{b,y}}
2	3	{{c,z}}

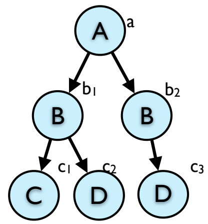
How-provenance for XML

- Consider *unordered XQuery*

```
p ::= l | $x | () | (p) | p,p | for $x in p return p
    | let $x := p return p | if (p=p) then p else p
    | element p {p} | name(p) | annot k p | p/s
s ::= ax::nt
ax ::= self | child | descendant
nt ::= l | *
```

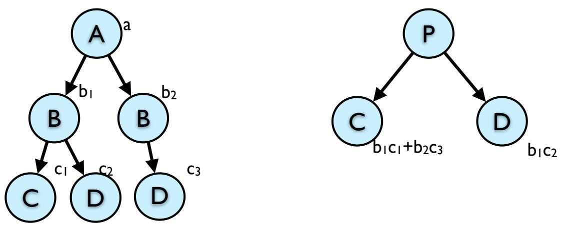
- Evaluate over *annotated (unordered) XML*
  - Each node of document has a semiring-valued annotation

Example




```
<p>{$doc/*/*}</p>
```


Example




```
<p>{$doc/*/*}</p>
```

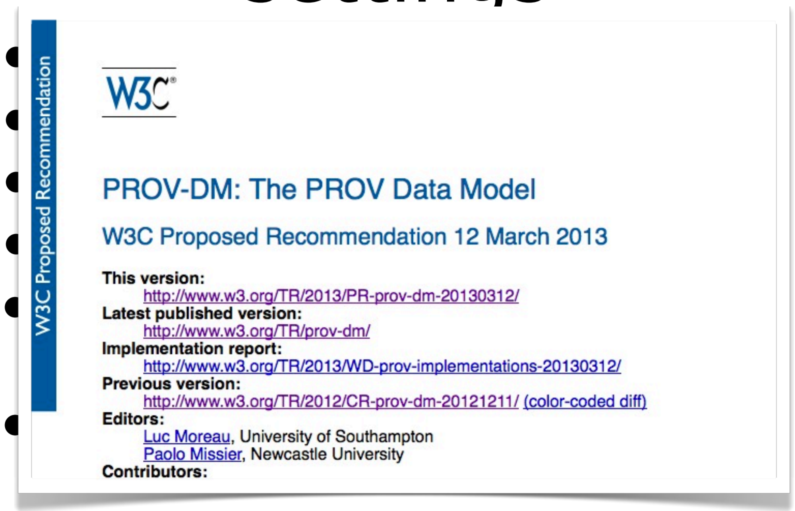

On the other hand...

- Semiring model is *not* the end of the story
- For example, where-provenance is not an instance of semiring model
  - There are other non-instances.
- Only handles *unordered* XML
  - also does not handle negation
- So, further generalization may be possible.

Provenance in other settings

- *Scientific workflows/distributed computing*
- *Business process modeling*
- *Semantic Web*
- *Operating systems, file systems*
- This work is generally not as formal
  - not as clear what is implemented and why
- Understanding and relating these models is important future work

Provenance in other settings



W3C Proposed Recommendation

W3C®

PROV-DM: The PROV Data Model

W3C Proposed Recommendation 12 March 2013

This version:
http://www.w3.org/TR/2013/PR-prov-dm-20130312/
Latest published version:
http://www.w3.org/TR/prov-dm/
Implementation report:
http://www.w3.org/TR/2013/WD-prov-implementations-20130312/
Previous version:
http://www.w3.org/TR/2012/CR-prov-dm-20121211/ (color-coded diff)
Editors:
Luc Moreau, University of Southampton
Paolo Missier, Newcastle University
Contributors:

Summary of course

- Standards/languages for XML
  - XPath/XQuery
  - XSLT
  - DTDs + XML Schema
- From XML to relations, and back
  - XML shredding
  - XML publishing

Summary of course

- Updates
  - XQuery Update
  - Updating XML stored in relations
- Types
  - Regular expression types/XDuce
  - XQuery typing, query/update independence
- Provenance - today

Presentations

- 10, 15, or 20 minutes (depending on group size)
- **Each group member must participate**
- Cover:
  - background
  - what you did (papers read, development)
  - status; experimental results
  - conclusions