Introduction to Fine-Grained Management of Data Provenance

Pierre Senellart

17 June 2021

Institut Pasteur

Provenance management

- Data management all about query evaluation

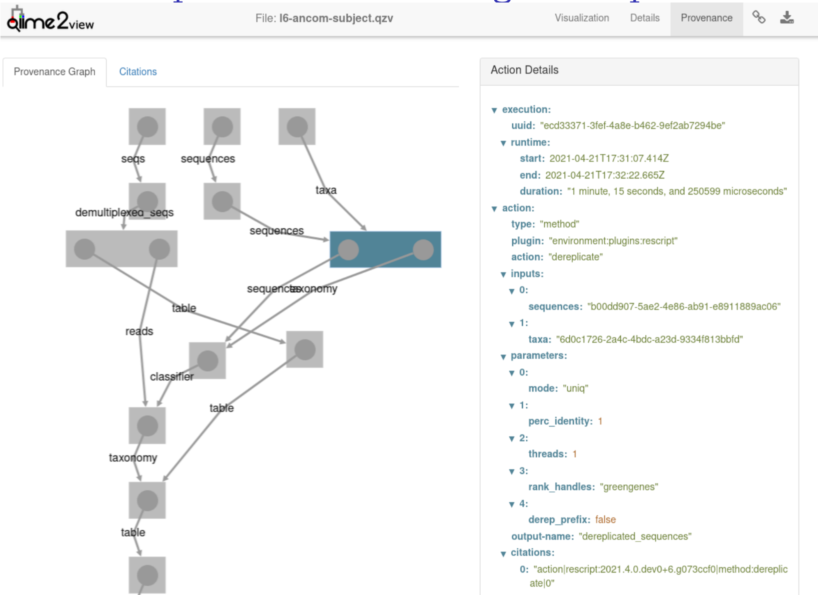
Provenance management

- Data management all about query evaluation
- What if we want something more than the query result?
  - Where does the result come from?
  - Why was this result obtained?
  - How was the result produced?
  - What is the probability of the result?
  - How many times was the result obtained?
  - How would the result change if part of the input data was missing?
  - What is the minimal security clearance I need to see the result?
  - What is the most economical way of obtaining the result?
  - How can a result be explained in layman terms?

Provenance management

- Data management all about query evaluation
- What if we want something more than the query result?
  - Where does the result come from?
  - Why was this result obtained?
  - How was the result produced?
  - What is the probability of the result?
  - How many times was the result obtained?
  - How would the result change if part of the input data was missing?
  - What is the minimal security clearance I need to see the result?
  - What is the most economical way of obtaining the result?
  - How can a result be explained in layman terms?
- Provenance management: along with query evaluation, record additional bookkeeping information allowing to answer the questions above

Workflow provenance vs fine-grained provenance



Workflow provenance vs fine-grained provenance

Workflow provenance

[Davidson et al., 2007]

- Uniquely identifies datasets used and produced
- Documents every action carried out (date, tool, version, parameters, inputs, outputs, etc.)
- Typically has a simple directed graph structure

Workflow provenance vs fine-grained provenance

Workflow provenance

[Davidson et al., 2007]

- Uniquely identifies datasets used and produced
- Documents every action carried out (date, tool, version, parameters, inputs, outputs, etc.)
- Typically has a simple directed graph structure

Data (fine-grained) provenance

[Buneman et al., 2001]

- At the level of a single data item (a record, a data value, a node in a graph, etc.)
- Documents how this particular data item was produced
- Possibly a rich mathematical structure
- Support for a limited set of data operations

Outline

Data management

Numerous applications (standalone software, Web sites, etc.) need to manage data:

- Structure data useful to the application
- Store them in a persistent manner (data retained even when the application is not running)
- Efficiently query information within large data volumes
- Update data without violating some structural constraints
- Enable data access and updates by multiple users, possibly concurrently

Often, desirable to access the same data from several distinct applications, from distinct computers.

Role of a DBMS

Database Management System

Software that simplifies the design of applications that handle data, by providing a unified access to the functionalities required for data management, whatever the application.

Database

Collection of data (specific to a given application) managed by a DBMS

Classical relational DBMSs

- Based on the relational model: decomposition of data into relations (i.e., tables)
- A standard query language: SQL
- An algebraic formulation of (a subset of) SQL, useful for reasoning and optimization: the relational algebra
- Data stored on disk
- Relations (tables) stored line after line
- Centralized system, with limited distribution possibilities

Example relational database

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Outline

The relational algebra

- Algebraic language to express queries
- A relational algebra expression produces a new relation from the database relations
- Each operator takes 0, 1, or 2 subexpressions
- Main operators:

Op.	Arity	Description	Condition
R	0	Relation name	$R \in \mathcal{L}$
$\rho_{A \to B}$	1	Renaming	$A, B \in \mathcal{L}$
$\Pi_{A_1 \ldots A_n}$	1	Projection	$A_1 \ldots A_n \in \mathcal{L}$
σ_φ	1	Selection	φ formula
$\times$	2	Cross product	
$\cup$	2	Union	
$\setminus$	2	Difference	
$\bowtie_\varphi$	2	Join	φ formula

Relation name

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression: `Guest`

Result:

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Renaming

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression: $\rho_{\texttt{id}\to\texttt{guest}}(\texttt{Guest})$

Result:

guest	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Projection

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression: $\Pi_{\mathtt{email,id}}(\mathtt{Guest})$

Result:

email	id
john.smith@gmail.com	1
alice@black.name	2
john.smith@ens.fr	3

Selection

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression: $\sigma_{\texttt{arrival}>2017\text{-}01\text{-}12 \wedge \texttt{guest}=2}(\texttt{Reservation})$

Result:

id	guest	room	arrival	nights
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

The formula used in the selection can be any Boolean combination of comparisons of attributes to attributes or constants.

Cross product

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression: $\Pi_{\texttt{id}}(\texttt{Guest}) \times \Pi_{\texttt{name}}(\texttt{Guest})$

Result:

id	name
1	Alice Black
2	Alice Black
3	Alice Black
1	John Smith
2	John Smith
3	John Smith

Union

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression: $\Pi_{\texttt{room}}(\sigma_{\texttt{guest}=2}(\texttt{Reservation})) \cup \Pi_{\texttt{room}}(\sigma_{\texttt{arrival}=\text{2017-01-15}}(\texttt{Reservation}))$

Result:

room
107
302
504

Union

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression: $\Pi_{\texttt{room}}(\sigma_{\texttt{guest}=2}(\texttt{Reservation})) \cup \Pi_{\texttt{room}}(\sigma_{\texttt{arrival}=2017\text{-}01\text{-}15}(\texttt{Reservation}))$

Result:

room
107
302
504

This simple union could have been written $\Pi_{\texttt{room}}(\sigma_{\texttt{guest}=2 \vee \texttt{arrival}=2017\text{-}01\text{-}15}(\texttt{Reservation}))$. Not always possible.

Difference

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression: $\Pi_{\texttt{room}}(\sigma_{\texttt{guest}=2}(\texttt{Reservation})) \setminus \Pi_{\texttt{room}}(\sigma_{\texttt{arrival}=2017\text{-}01\text{-}15}(\texttt{Reservation}))$

Result:

room
107

Difference

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression: $\Pi_{\texttt{room}}(\sigma_{\texttt{guest}=2}(\texttt{Reservation})) \setminus \Pi_{\texttt{room}}(\sigma_{\texttt{arrival}=2017\text{-}01\text{-}15}(\texttt{Reservation}))$

Result:

room
107

This simple difference could have been written $\Pi_{\texttt{room}}(\sigma_{\texttt{guest}=2 \land \texttt{arrival} \neq 2017\text{-}01\text{-}15}(\texttt{Reservation}))$. Not always possible.

Join

Guest

id	name	email
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Reservation

id	guest	room	arrival	nights
1	1	504	2017-01-01	5
2	2	107	2017-01-10	3
3	3	302	2017-01-15	6
4	2	504	2017-01-15	2
5	2	107	2017-01-30	1

Expression: $\mathtt{Reservation} \bowtie_{\mathtt{guest}=\mathtt{id}} \mathtt{Guest}$

Result:

id	guest	room	arrival	nights	name	email
1	1	504	2017-01-01	5	John Smith	john.smith@gmail.com
2	2	107	2017-01-10	3	Alice Black	alice@black.name
3	3	302	2017-01-15	6	John Smith	john.smith@ens.fr
4	2	504	2017-01-15	2	Alice Black	alice@black.name
5	2	107	2017-01-30	1	Alice Black	alice@black.name

The formula used in the join can be any Boolean combination of comparisons of attributes of the table on the left to attributes of the table on the right.

Note on the join

- The join is not an elementary operator of the relational algebra (but it is very useful)
- It can be seen as a combination of renaming, cross product, selection, projection
- Thus:

$$\begin{aligned} &\texttt{Reservation} \bowtie_{\texttt{guest=id}} \texttt{Guest} \\ \equiv\ &\Pi_{\texttt{id,guest,room,arrival,nights,name,email}}(\\ &\quad \sigma_{\texttt{guest=temp}}(\texttt{Reservation} \times \rho_{\texttt{id}\to\texttt{temp}}(\texttt{Guest}))) \end{aligned}$$

- If R and S have for attributes $\mathcal{A}$ and $\mathcal{B}$, we note $R \bowtie S$ the natural join of R and S, where the join formula is $\bigwedge_{A\in\mathcal{A}\cap\mathcal{B}} A = A$.

Outline

Data model

- Relational data model: data decomposed into relations, with labeled attributes...

Data model

- Relational data model: data decomposed into relations, with labeled attributes…

name	position	city	classification
John	Director	New York	unclassified
Paul	Janitor	New York	restricted
Dave	Analyst	Paris	confidential
Ellen	Field agent	Berlin	secret
Magdalen	Double agent	Paris	top secret
Nancy	HR director	Paris	restricted
Susan	Analyst	Berlin	secret

Data model

- Relational data model: data decomposed into relations, with labeled attributes...
- ... with an extra provenance annotation for each tuple (think of it first as a tuple id)

name	position	city	classification	prov
John	Director	New York	unclassified	t_1
Paul	Janitor	New York	restricted	t_2
Dave	Analyst	Paris	confidential	t_3
Ellen	Field agent	Berlin	secret	t_4
Magdalen	Double agent	Paris	top secret	t_5
Nancy	HR director	Paris	restricted	t_6
Susan	Analyst	Berlin	secret	t_7

Relations and databases

Formally:

- A relational schema $\mathcal{R}$ is a finite sequence of distinct attribute names; the arity of $\mathcal{R}$ is $|\mathcal{R}|$
- A database schema is a mapping from relation names to relational schemas, with finite support
- A tuple over relation schema $\mathcal{R}$ is a mapping from $\mathcal{R}$ to data values; each tuple comes with a provenance annotation
- A relation instance (or relation) over $\mathcal{R}$ is a finite set of tuples over $\mathcal{R}$
- A database instance (or database) over database schema $\mathcal{D}$ is a mapping from the support of $\mathcal{D}$ mapping each relation name R to a relation instance over $\mathcal{D}(R)$

Queries

- A query is an arbitrary function that maps databases over a fixed database schema $\mathcal{D}$ to relations over some relational schema $\mathcal{R}$
- The query does not consider or produce any provenance annotations; we will give semantics for the provenance annotations of the output, based on that of the input
- In practice, one often restricts to specific query languages:
  - Monadic-Second Order logic (MSO)
  - First-Order logic (FO) or the relational algebra
  - SQL with aggregate functions
  - etc.

Outline

Boolean provenance [Imieliński and Lipski, 1984]

- $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ finite set of Boolean events
- Provenance annotation: Boolean function over $\mathcal{X}$, i.e., a function of the form: $(\mathcal{X} \to \{\bot, \top\}) \to \{\bot, \top\}$
- Interpretation: possible-world semantics
  - every valuation $\nu : \mathcal{X} \to \{\bot, \top\}$ denotes a possible world of the database
  - the provenance of a tuple on ν evaluates to $\bot$ or $\top$ depending whether this tuple exists in that possible world
  - for example, if every tuple of a database is annotated with the indicator function of a distinct Boolean event, the set of possible worlds is the set of all subdatabases

Example of possible worlds

name	position	city	classification	prov
John	Director	New York	unclassified	t_1
Paul	Janitor	New York	restricted	t_2
Dave	Analyst	Paris	confidential	t_3
Ellen	Field agent	Berlin	secret	t_4
Magdalen	Double agent	Paris	top secret	t_5
Nancy	HR director	Paris	restricted	t_6
Susan	Analyst	Berlin	secret	t_7

$$\nu: \quad \begin{array}{ccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \\ \top & \top & \top & \top & \top & \top & \top \end{array}$$

Example of possible worlds

name	position	city	classification	prov
John	Director	New York	unclassified	t_1
Dave	Analyst	Paris	confidential	t_3
Magdalen	Double agent	Paris	top secret	t_5
Susan	Analyst	Berlin	secret	t_7

$$\nu: \quad \begin{array}{ccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \\ \top & \bot & \top & \bot & \top & \bot & \top \end{array}$$

Boolean provenance of query results

- $\nu(D)$: the subdatabase of D where all tuples whose provenance annotation evaluates to $\perp$ by ν are removed
- The Boolean provenance $\mathrm{prov}_{q,D}(t)$ of tuple $t \in q(D)$ is the function:

$$\nu \mapsto \begin{cases} \top \text{ if } t \in q(\nu(D)) \\ \perp \text{ otherwise} \end{cases}$$

Example (What cities are in the table?)

name	position	city	classification	prov
John	Director	New York	unclassified	t_1
Paul	Janitor	New York	restricted	t_2
Dave	Analyst	Paris	confidential	t_3
Ellen	Field agent	Berlin	secret	t_4
Magdalen	Double agent	Paris	top secret	t_5
Nancy	HR director	Paris	restricted	t_6
Susan	Analyst	Berlin	secret	t_7

city	prov
New York	$t_1 \vee t_2$
Paris	$t_3 \vee t_5 \vee t_6$
Berlin	$t_4 \vee t_7$

What now?

- How to compute Boolean provenance for practical query languages? What complexity?
- What can we do with provenance?
- How to use provenance in practice?

Outline

Commutative semiring $(K, \mathbb{0}, \mathbb{1}, \oplus, \otimes)$

- Set K with distinguished elements $\mathbb{0}$, $\mathbb{1}$
- $\oplus$ associative, commutative operator, with identity $\mathbb{0}_K$:
  - $a \oplus (b \oplus c) = (a \oplus b) \oplus c$
  - $a \oplus b = b \oplus a$
  - $a \oplus \mathbb{0} = \mathbb{0} \oplus a = a$
- $\otimes$ associative, commutative operator, with identity $\mathbb{1}_K$:
  - $a \otimes (b \otimes c) = (a \otimes b) \otimes c$
  - $a \otimes b = b \otimes a$
  - $a \otimes \mathbb{1} = \mathbb{1} \otimes a = a$
- $\otimes$ distributes over $\oplus$:

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$$

- $\mathbb{0}$ is annihilating for $\otimes$:

$$a \otimes \mathbb{0} = \mathbb{0} \otimes a = \mathbb{0}$$

Example semirings

- $(\mathbb{N}, 0, 1, +, \times)$: counting semiring
- $(\{\bot, \top\}, \bot, \top, \vee, \wedge)$: Boolean semiring
- $(\{\mathit{unclassified}, \mathit{restricted}, \mathit{confidential}, \mathit{secret}, \mathit{top\ secret}\}, \mathit{top\ secret}, \mathit{unclassified}, \min, \max)$: security semiring
- $(\mathbb{N} \cup \{\infty\}, \infty, 0, \min, +)$: tropical semiring
- $(\{\text{Boolean functions over } \mathcal{X}\}, \bot, \top, \vee, \wedge)$: semiring of Boolean functions over $\mathcal{X}$
- $(\mathbb{N}[\mathcal{X}], 0, 1, +, \times)$: semiring of integer-valued polynomials with variables in $\mathcal{X}$ (also called How-semiring or universal semiring)
- $(\mathcal{P}(\mathcal{P}(\mathcal{X})), \emptyset, \{\emptyset\}, \cup, \Cup)$: Why-semiring over $\mathcal{X}$ $(A \Cup B := \{a \cup b \mid a \in A, b \in B\})$

Semiring provenance [Green et al., 2007]

- We fix a semiring $(K, \mathbb{0}, \mathbb{1}, \oplus, \otimes)$
- We assume provenance annotations are in K
- We consider a query q from the positive relational algebra (selection, projection, renaming, cross product, union; joins can be simulated with renaming, cross product, selection, projection)
- We define a semantics for the provenance of a tuple $t \in q(D)$ inductively on the structure of q

Selection, renaming

Provenance annotations of selected tuples are unchanged

Example ($\rho_{\text{name}\to\text{n}}(\sigma_{\text{city}=\text{"New York"}}(R))$)

name	position	city	classification	prov
John	Director	New York	unclassified	t_1
Paul	Janitor	New York	restricted	t_2
Dave	Analyst	Paris	confidential	t_3
Ellen	Field agent	Berlin	secret	t_4
Magdalen	Double agent	Paris	top secret	t_5
Nancy	HR director	Paris	restricted	t_6
Susan	Analyst	Berlin	secret	t_7

n	position	city	classification	prov
John	Director	New York	unclassified	t_1
Paul	Janitor	New York	restricted	t_2

Projection

Provenance annotations of identical, merged, tuples are $\oplus$-ed

Example ($\pi_{\text{city}}(R)$)

name	position	city	classification	prov
John	Director	New York	unclassified	t_1
Paul	Janitor	New York	restricted	t_2
Dave	Analyst	Paris	confidential	t_3
Ellen	Field agent	Berlin	secret	t_4
Magdalen	Double agent	Paris	top secret	t_5
Nancy	HR director	Paris	restricted	t_6
Susan	Analyst	Berlin	secret	t_7

city	prov
New York	$t_1 \oplus t_2$
Paris	$t_3 \oplus t_5 \oplus t_6$
Berlin	$t_4 \oplus t_7$

Union

Provenance annotations of identical, merged, tuples are $\oplus$-ed

Example

$$\pi_{\text{city}}(\sigma_{\textit{ends-with}(\text{position},\text{"agent"})}(R)) \cup \pi_{\text{city}}(\sigma_{\text{position}=\text{"Analyst"}}(R))$$

name	position	city	classification	prov
John	Director	New York	unclassified	t_1
Paul	Janitor	New York	restricted	t_2
Dave	Analyst	Paris	confidential	t_3
Ellen	Field agent	Berlin	secret	t_4
Magdalen	Double agent	Paris	top secret	t_5
Nancy	HR director	Paris	restricted	t_6
Susan	Analyst	Berlin	secret	t_7

city	prov
Paris	$t_3 \oplus t_5$
Berlin	$t_4 \oplus t_7$

Cross product

Provenance annotations of combined tuples are $\otimes$-ed

Example

$$\pi_{\text{city}}(\sigma_{\textit{ends-with}(\text{position},\text{"agent"})}(R)) \bowtie \pi_{\text{city}}(\sigma_{\text{position}=\text{"Analyst"}}(R))$$

name	position	city	classification	prov
John	Director	New York	unclassified	t_1
Paul	Janitor	New York	restricted	t_2
Dave	Analyst	Paris	confidential	t_3
Ellen	Field agent	Berlin	secret	t_4
Magdalen	Double agent	Paris	top secret	t_5
Nancy	HR director	Paris	restricted	t_6
Susan	Analyst	Berlin	secret	t_7

city	prov
Paris	$t_3 \otimes t_5$
Berlin	$t_4 \otimes t_7$

What can we do with it?

counting semiring: count the number of times a tuple can be derived, multiset semantics

Boolean semiring: determines if a tuple exists when a subdatabase is selected

security semiring: determines the minimum clearance level required to get a tuple as a result

tropical semiring: minimum-weight way of deriving a tuple (think shortest path in a graph)

Boolean functions: Boolean provenance, as previously defined

integer polynomials: universal provenance, see further

Why-semiring: Why-provenance [Buneman et al., 2001], set of combinations of tuples needed for a tuple to exist

Example of security provenance

$$\pi_{\text{city}}(\sigma_{\text{name}<\text{name2}}(\pi_{\text{name,city}}(R) \bowtie \rho_{\text{name}\to\text{name2}}(\pi_{\text{name,city}}(R))))$$

name	position	city	prov
John	Director	New York	unclassified
Paul	Janitor	New York	restricted
Dave	Analyst	Paris	confidential
Ellen	Field agent	Berlin	secret
Magdalen	Double agent	Paris	top secret
Nancy	HR director	Paris	restricted
Susan	Analyst	Berlin	secret

city	prov
New York	restricted
Paris	confidential
Berlin	secret

Outline

Application: Probabilistic databases

[Green and Tannen, 2006, Suciu et al., 2011]

- Tuple-independent database: each tuple t in a database is annotated with independent probability $\Pr(t)$ of existing
- Probability of a possible world $D' \subseteq D$:

$$\Pr(D') = \prod_{t \in D'} \Pr(t) \times \prod_{t \in D' \setminus D} (1 - \Pr(t'))$$

- Probability of a tuple for a query q over D:

$$\Pr(t \in q(D)) = \sum_{\substack{D' \subseteq D \\ t \in q(D')}} \Pr(D')$$

- If $\Pr(x_i) := \Pr(t_i)$ where x_i is the provenance annotation of tuple t_i then $\Pr(t \in q(D)) = \Pr(\mathrm{prov}_{q,D}(t))$
- Computing the probability of a query in probabilistic databases thus amounts to computing Boolean provenance, and then computing the probability of a Boolean function
- Also works for more complex probabilistic models

Example of probability computation

name	position	city	classification	prov	prob
John	Director	New York	unclassified	t_1	0.5
Paul	Janitor	New York	restricted	t_2	0.7
Dave	Analyst	Paris	confidential	t_3	0.3
Ellen	Field agent	Berlin	secret	t_4	0.2
Magdalen	Double agent	Paris	top secret	t_5	1.0
Nancy	HR director	Paris	restricted	t_6	0.8
Susan	Analyst	Berlin	secret	t_7	0.2

city	prov
New York	$t_1 \vee t_2$
Paris	$t_3 \vee t_5 \vee t_6$
Berlin	$t_4 \vee t_7$

Example of probability computation

name	position	city	classification	prov	prob
John	Director	New York	unclassified	t_1	0.5
Paul	Janitor	New York	restricted	t_2	0.7
Dave	Analyst	Paris	confidential	t_3	0.3
Ellen	Field agent	Berlin	secret	t_4	0.2
Magdalen	Double agent	Paris	top secret	t_5	1.0
Nancy	HR director	Paris	restricted	t_6	0.8
Susan	Analyst	Berlin	secret	t_7	0.2

city	prov	prob
New York	$t_1 \vee t_2$	$1-(1-0.5)\times(1-0.7)=0.85$
Paris	$t_3 \vee t_5 \vee t_6$	1.00
Berlin	$t_4 \vee t_7$	$1-(1-0.2)\times(1-0.2)=0.36$

Outline

Preliminaries

Provenance

Applications

- Probabilistic databases
- Explanation

Conclusion

Using provenance for explanation

- Semiring provenance can be used to provide a user with explanation on the query result:
  - How-provenance (provenance polynomials) explains precisely how a result has been computed: often too fine-grained
  - Why-provenance explains why a particular result is generated by providing combinations of tuples required for a tuple to be produced
- Provenance often too long and complex, (imperfect) summarization may be required [Ainy et al., 2015]
- Still far from a natural language explanation!
- Why-not provenance: why a result was not produced. Expressible with m-semirings, but requires dedicated techniques [Chapman and Jagadish, 2009] for compact explanations

ProvSQL: Provenance within PostgreSQL (1/2)

[Senellart et al., 2018]

- Lightweight extension/plugin for PostgreSQL ≥ 9.5
- Provenance annotations stored as UUIDs, in an extra attribute of each provenance-aware relation
- A provenance circuit relating UUIDs of elementary provenance annotations and arithmetic gates stored as table
- All computations done in the universal semiring (more precisely, extensions of it to support more operations)
- Probability computation from the provenance circuits, via various methods

ProvSQL: Current status

- Supported SQL language features:
  - Regular SELECT-FROM-WHERE queries (aka conjunctive queries with multiset semantics)
  - JOIN queries (regular joins and outer joins; semijoins and antijoins are not currently supported)
  - SELECT queries with nested SELECT subqueries in the FROM clause
  - GROUP BY queries (without aggregation)
  - SELECT DISTINCT queries (i.e., set semantics)
  - UNION's or UNION ALL's of SELECT queries
  - EXCEPT queries
  - Final aggregate (COUNT, MIN, SUM, etc.) queries
- Try it (see a demo, do the tutorial) from https://github.com/PierreSenellart/provsql

Other databases with provenance management

- Older probabilistic database systems can compute some forms of provenance (especially, Boolean provenance); but tied to a specific version of PostgreSQL, hard to deploy

  Trio: http://infolab.stanford.edu/trio/ [Benjelloun et al., 2006]

  MayBMS: http://maybms.sourceforge.net/ [Huang et al., 2009]

- Perm https://github.com/IITDBGroup/perm [Glavic and Alonso, 2009] now obsolete system for provenance management; also tied to a specific version of PostgreSQL
- GProM http://www.cs.iit.edu/~dbgroup/projects/gprom.html [Arab et al., 2018] is similar to ProvSQL (though no probabilistic database capabilities), with some extra features; implemented as a middleware

In brief and beyond. . . [Senellart, 2017, 2019]

- Quite rich foundations of provenance management:
  - Different types of provenance
  - Semiring formalism to unify most provenance forms
  - (Partial) extensions for difference, recursive queries, aggregation, updates; to other data models
  - Compact provenance representation formalisms
- Now is the time to work on concrete, efficient, usable implementation (my job!)
- Now is the time to work with actual users, to adapt to actual needs of users who want to track the provenance of the data at a fine-grained level!

Merci.

`https://github.com/PierreSenellart/provsql`

`https://youtu.be/iqzSNfGHbEE?vq=hd1080`

Bibliography I

Eleanor Ainy, Pierre Bourhis, Susan B. Davidson, Daniel Deutch, and Tova Milo. Approximated summarization of data provenance. In *CIKM*, 2015.

Bahareh Sadat Arab, Su Feng, Boris Glavic, Seokki Lee, Xing Niu, and Qitian Zeng. GProM - A swiss army knife for your provenance needs. *IEEE Data Eng. Bull.*, 41(1):51–62, 2018.

Omar Benjelloun, Anish Das Sarma, Alon Halevy, and Jennifer Widom. ULDBs: Databases with uncertainty and lineage. In *VLDB*, pages 953–964, 2006.

Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. Why and where: A characterization of data provenance. In *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings.*, 2001.

Bibliography II

Adriane Chapman and H. V. Jagadish. Why not? In *SIGMOD*, 2009.

Susan B. Davidson, Sarah Cohen Boulakia, Anat Eyal, Bertram Ludäscher, Timothy M. McPhillips, Shawn Bowers, Manish Kumar Anand, and Juliana Freire. Provenance in scientific workflow systems. *IEEE Data Eng. Bull.*, 30(4): 44–50, 2007. URL `http://sites.computer.org/debull/A07dec/susan.pdf`.

Boris Glavic and Gustavo Alonso. Perm: Processing provenance and data on the same data model through query rewriting. In *ICDE*, pages 174–185, 2009.

Todd J. Green and Val Tannen. Models for incomplete and probabilistic information. *IEEE Data Eng. Bull.*, 29(1), 2006.

Bibliography III

Todd J Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *PODS*, 2007.

Jiewen Huang, Lyublena Antova, Christoph Koch, and Dan Olteanu. MayBMS: a probabilistic database management system. In *SIGMOD*, pages 1071–1074, 2009.

Tomasz Imieliński and Jr. Lipski, Witold. Incomplete information in relational databases. *J. ACM*, 31(4), 1984.

Pierre Senellart. Provenance and probabilities in relational databases: From theory to practice. *SIGMOD Record*, 46(4), December 2017.

Pierre Senellart. Provenance in Databases: Principles and Applications. In *Proc. RW*, pages 104–109, Bolzano, Italy, September 2019.

Bibliography IV

Pierre Senellart, Louis Jachiet, Silviu Maniu, and Yann Ramusat. ProvSQL: provenance and probability management in postgresql. In *VLDB*, 2018. Demonstration.

Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*. Morgan & Claypool, 2011.