A Conceptual Model and Predicate Language for Data Selection and Projection Based on Provenance

David W. Archer and Lois M. L. Delcambre

Department of Computer Science

Portland State University

Topics

- Motivation
- Conceptual Model
- Predicate Language
- Evaluation

Data Curation Settings

- Fine-grained data from multiple sources
- Integrated, queried, and further updated or manipulated
  - Evolving schema and instance
  - Multiple histories that include manipulations and queries
  - Multiple values for attributes
  - User expressions of confidence and doubt
- Example Settings
  - Intelligence: profiling “persons of interest”
  - Military: operation risk assessment
  - eScience: Bioinformatics databases

When is Curated Data Trustworthy?

Name	ID
Bob	8, 9
Sue	7

- → Do we trust the people that derived it?
- → Do we trust how and in what order it was derived?
- → Do we know which source(s)* data came from?
- → If processing methods were used to derive the data, have they improved or changed?

Where Current Models Fall Short,1

- Provenance is limited
  - Single history
  - Single granularity (mostly)
  - Query or DML, but not both (mostly)
- Some models store provenance in the same schema as the data
  - Annotations stored as extra attributes
  - Creates “clutter”, and requires special care to prevent corruption during queries

Where Current Models Fall Short, 2

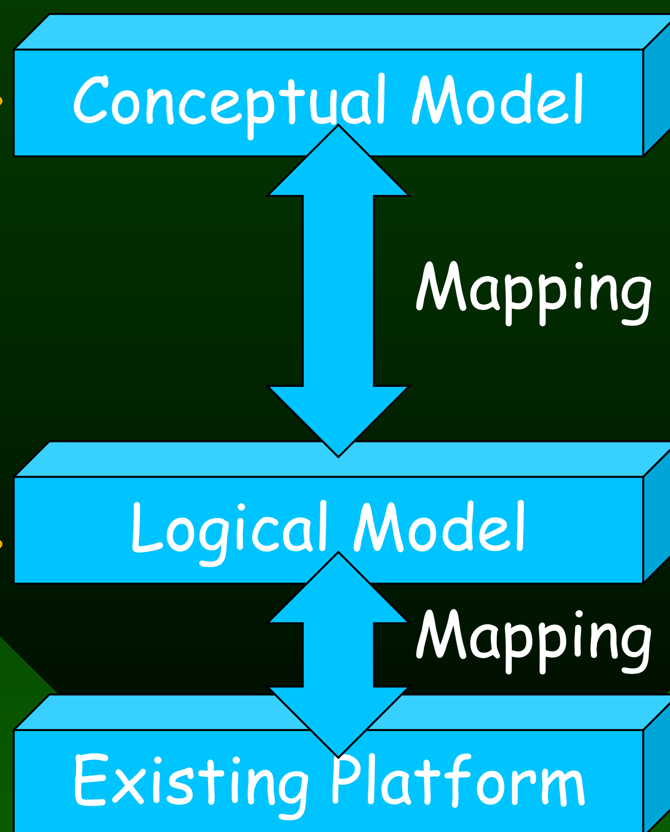
- Provenance stored as string annotations to data, so queries about provenance must parse the strings used by a particular system
- Provenance stored “one generation at a time”, so queries must be written recursively, to trace provenance through multiple prior queries

- Motivation
- Conceptual Model
- Predicate Language
- Evaluation

Overview of Our Research

- User view of data, provenance
- Simple, familiar language
- Data and prov. accessible
- Track provenance, but keep management of it out of user's hands

- Transition layer to implementations
- Performance
- Full access to provenance

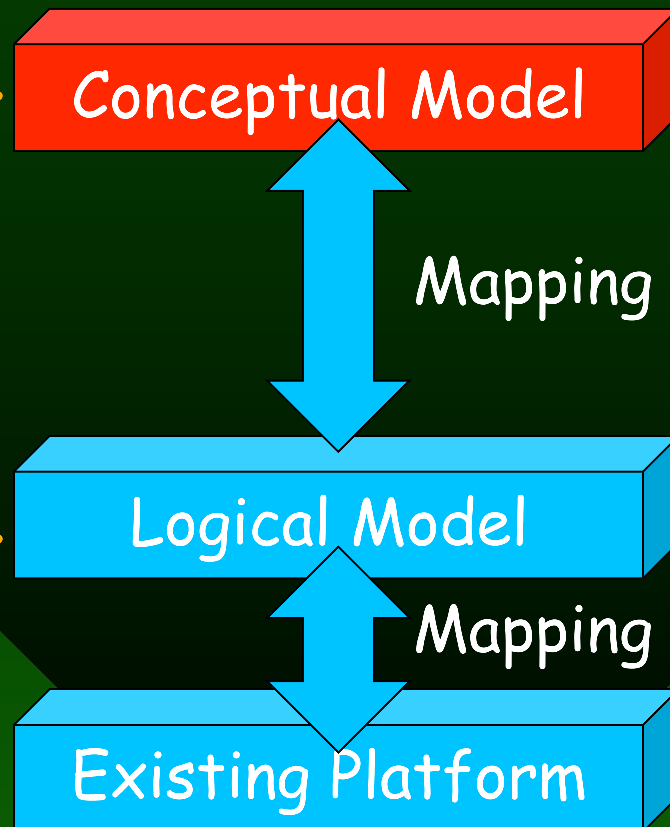


Overview of Our Research

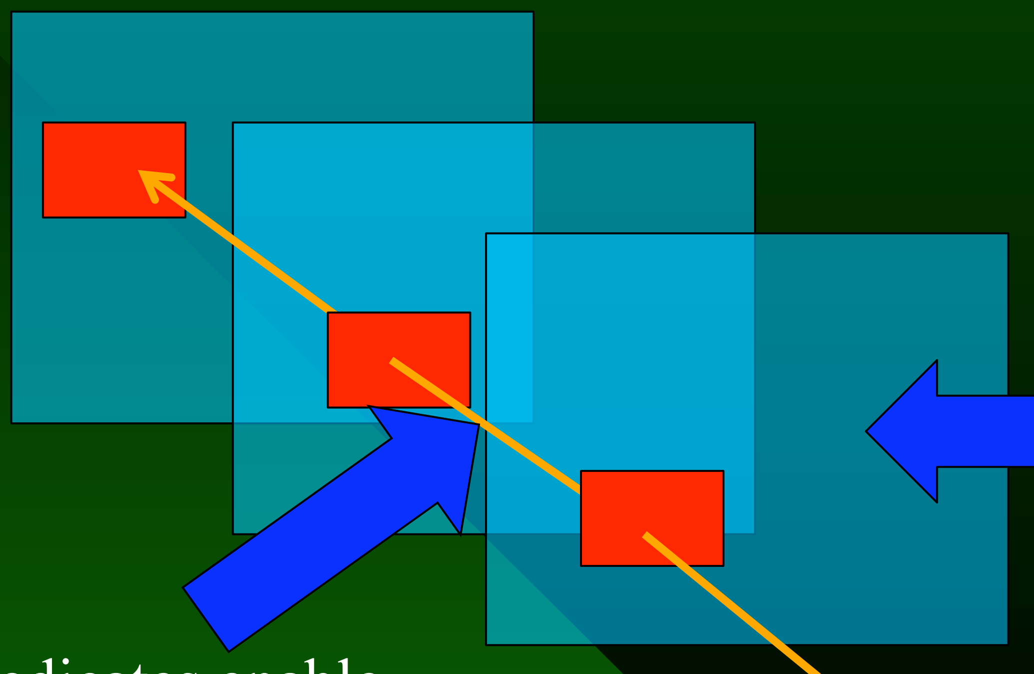
- User view of data, provenance
- Simple, familiar language
- Data and prov. accessible
- Track provenance, but keep management of it out of user's hands

- Transition layer to implementations
- Performance
- Full access to provenance

Focus of this paper



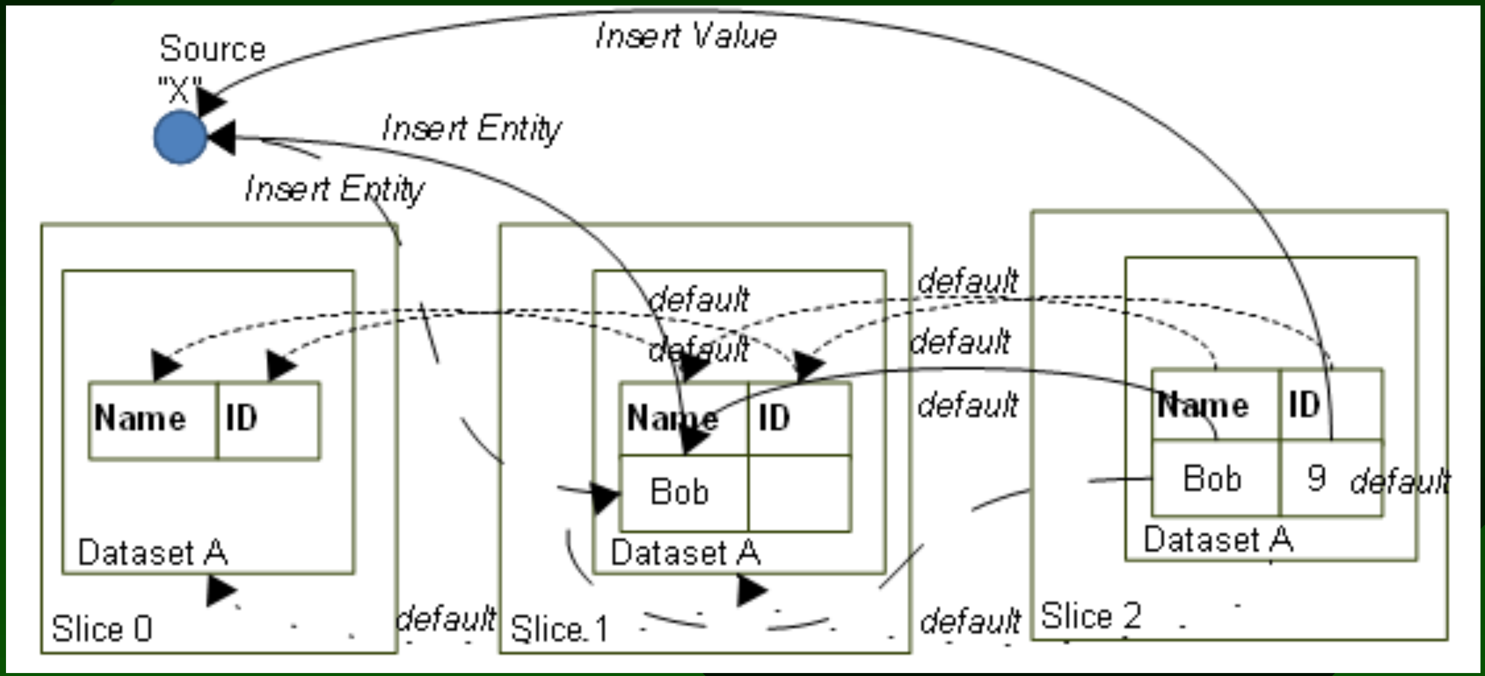
Idea: New predicates, not a new, full-featured provenance query language

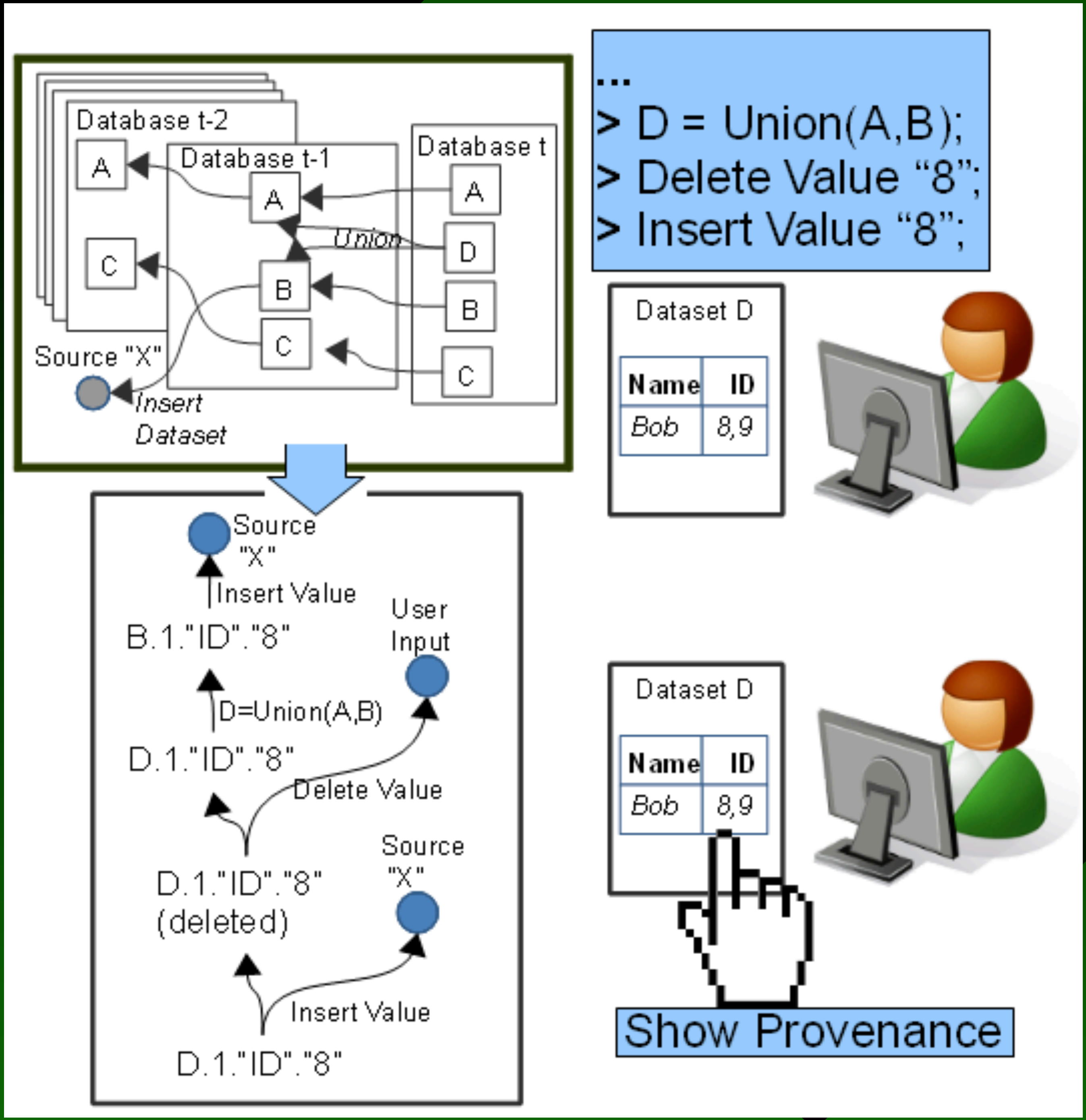


Normal relational algebra operates on “front face”

New predicates enable selection and projection based on provenance

Conceptual Model Structures




Database t-2
A
C
Database t-1
A
Union
B
C
Database t
A
D
B
C
Source "X"
Insert
Dataset
...
> D = Union(A,B);
> Delete Value "8";
> Insert Value "8";
Dataset D
Name
ID
Bob
8,9
Source
"X"
Insert Value
B.1."ID"."8"
User
Input
D=Union(A,B)
D.1."ID"."8"
Delete Value
Source
"X"
D.1."ID"."8"
(deleted)
Insert Value
D.1."ID"."8"
Dataset D
Name
ID
Bob
8,9
Show Provenance

Key Conceptual Model Features

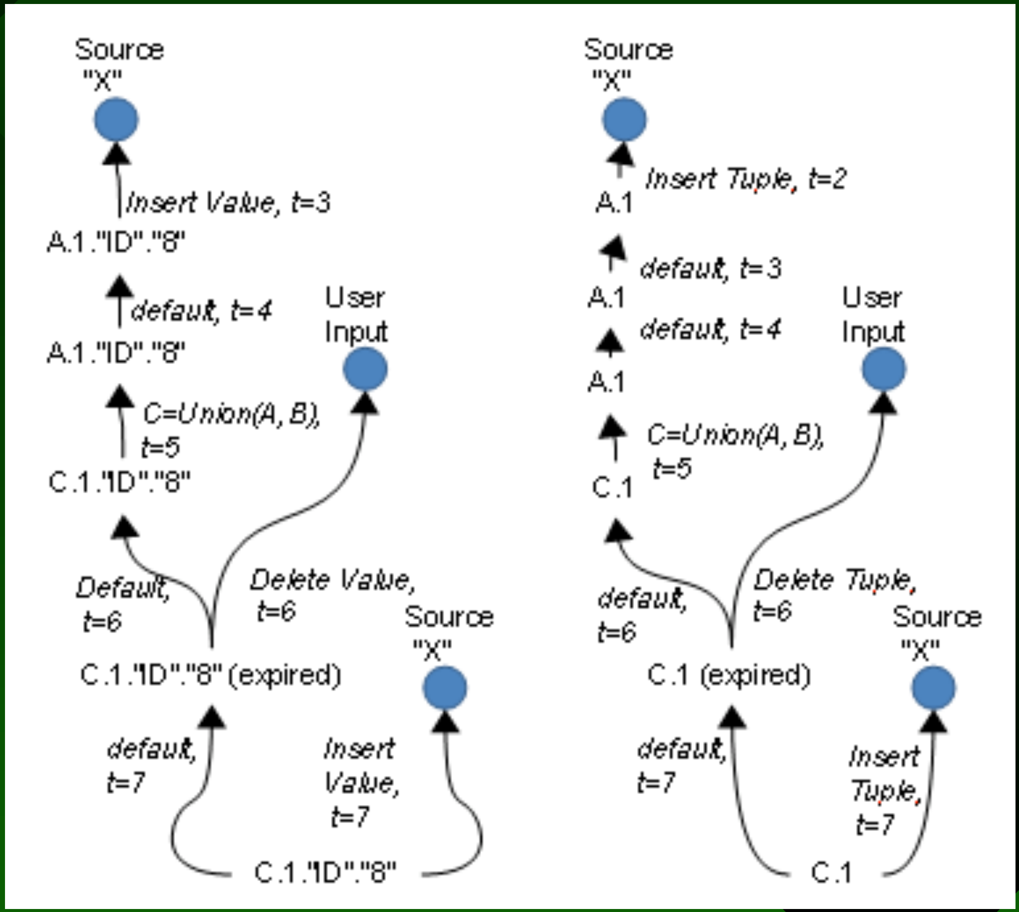
- Relational data with multi-valued attributes
- *Multi-layer multi-provenance* for all operations
  - Queries + DML + DDL
  - Data confidence language (DCL)
  - Distinct provenance for datasets, attributes, entities, and values
  - Deleted data and its provenance retained, re-insertions connected to prior deletions
  - Multiple histories for data

- Motivation
- Conceptual Model
- Predicate Language
- Evaluation

Simple Provenance Queries

- Goal: Enable selection of data by provenance
- Approach: predicate language for describing characteristics of provenance *paths* for both Select and Project operators
- Declarative, not procedural

Starting Point: Provenance Graphs



Predicate Language 1

selectionPredicate ::= TUPLE HAS <predicateQualifier> |
SOME DATA VALUE IN TUPLE HAS <predicateQualifier> |
A VALUE FROM ATTRIBUTES {list} IN TUPLE HAS <predicateQualifier>

projectionPredicate ::= ATTRIBUTE HAS <predicateQualifier> |
SOME DATA VALUE IN ATTRIBUTE HAS <predicateQualifier>

predicateQualifier ::= A PATH WITH (<pathQualifier>) |
A PATH WITH (<pathQualifier>) [AND|OR] <predicateQualifier>

pathQualifier ::= A <component>* (<cQualSet>) |
AN OPERATION (<aQualSet>) |
A SOURCE (<sQualSet) |
NOT <pathQualifier> |
<pathQualifier> [BEFORE|AND|OR] <pathQualifier>

* must agree with the component type specified in the selectionPredicate or projectionPredicate

Predicate Language 2

aQualSet ::= <aQual> | <aQual> [AND|OR] <aQualSet>

cQualSet ::= <cQual> | <cQual> [AND|OR] <cQualSet>

sQualSet ::= <sQual> | <sQual> [AND|OR] <sQualSet>

aQual ::= WITH ACTION = <constant> | WITH ACTION = A QUERY |

BY USER = <constant> | WHERE TIME <cCmp> <constant>

cQual ::= IN DATASET <cCmp> <constant> | WITH A VALUE <cCmp> <constant> |

THAT IS EXPIRED

sQual ::= WITH NAME <cCmp> <constant>

component ::= tuple | attribute | value

cCmp ::= = | > | < | ≥ | ≤ | ≠

Example Queries

Which tuples in relation R were derived from source "X”?

SELECT *
FROM R
WHERE (tuple has a path with (a source with name = “X”))

Which tuples in R have at least one data value derived from relation
"A" or relation "B”?

SELECT *
FROM R
WHERE (some data value in tuple has
a path with (a value in relation = "A”)
or a path with (a value in relation = "B”))

Which tuples contain data derived from relation "A" that later appeared in relation "C"?

```
SELECT *
FROM R
WHERE (some data value in tuple has a path with
        (a value in relation = "A"
        before a value in relation = "C"))
```

Which tuples are derived from tuples that were inserted at least once between timestamps "4" and "7"?

```
SELECT *
FROM R
WHERE (tuple has a path with (an operation with action =
"INSERT"  and where time >= "4" and where time < "7"))
```

- Motivation
- Conceptual Model
- Predicate Language
- Evaluation

MMP and Trio Provenance Selection Languages Compared

Kind of component being selected

Kind of ancestry to select by

Complexity of path conditions

	Data	Actions	Data and Actions	
Entities	√	√	√	Single
	√	√	√	Multiple, unordered
	√	√	√	Multiple, ordered
Attributes	√	√	√	Single
	√	√	√	Multiple, unordered
	√	√	√	Multiple, ordered

Our predicate language

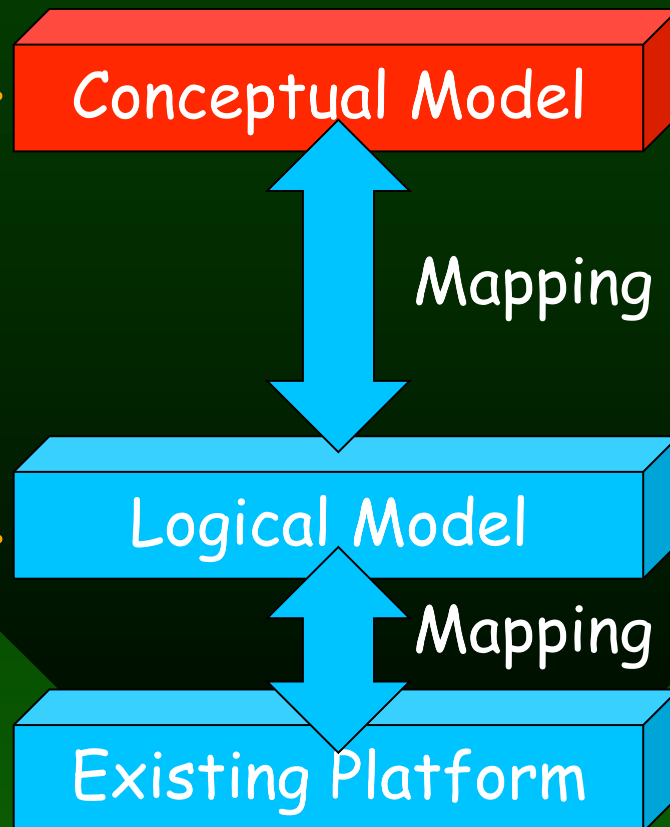
	Data	Actions	Data and Actions	
Entities	√			Single
	√			Multiple, unordered
				Multiple, ordered
Attributes				Single
				Multiple, unordered
				Multiple, ordered

Trio's predicate language, with Lineage()

Overview of Our Research

- User view of data, provenance
- Simple, familiar language
- Data and prov. accessible
- Track provenance, but keep management of it out of user's hands

Focus of this paper



- Transition layer to implementations
- Performance
- Full access to provenance

Implementation Feasibility

- Identify provenance graphs to search
  - As with all operations, starting point is Now
  - Query specifies input relation
  - Predicate specifies tuples, attributes, or values
- Encode predicate as GraphQL patterns
- Tuples or attributes selected for output if at least one relevant provenance graph is selected by GraphQL

Work in Progress

- Conceptual model
  - Formalization of subset in algebraic structure
  - Comparing expressiveness
  - Comparing query complexity
  - Closure and other properties
- Proof of Inter-model mapping
- Logical model
  - Open-ended access via other query languages
  - Implementation feasibility
  - Performance trade-off studies

Backup Material

Summary of MMP Differences

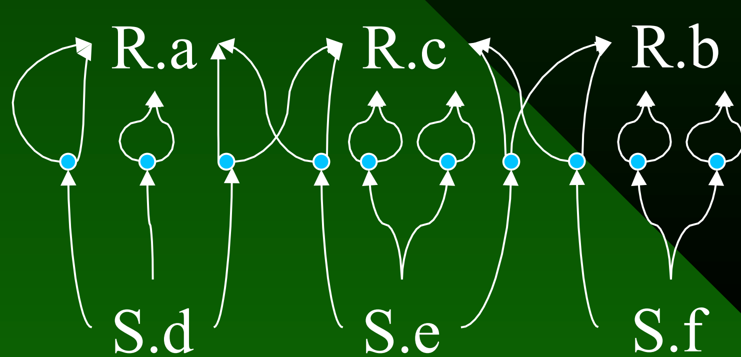
Data structure	Simple non-first normal relational
Orthogonal provenance and data?	Yes
Multi-generation provenance?	Yes
Multi-granularity provenance?	Yes
Multi-history provenance?	Yes
Operators	DDL, DML, Query, Confirm/Doubt
Deleted data provenanced?	Yes
Re-insertions connected?	Yes
Language to extract provenance?	In logical model
Simple language to select data based on provenance?	In conceptual model

Provenance Representations

Tuple ID	A	B	C
a	1	5	8
b	3	2	9
c	1	6	9

$S = \pi_{AC}(R\ (_A\ R) \cup (R\ (_C\ R)$

S		Provenance Representations			
A	C	Lineage	Why	Trio	Green
d. 1	8	{a,c}	{{a},{a,c}}	$2a + ac$	$2a^2 + ac$
e. 1	9	{a,b,c}	{{c},{a,c},{b,c}}	$2c + ac + bc$	$2c^2 + ac + bc$
f. 3	9	{b,c}	{{b},{b,c}}	$2b + bc$	$2b^2 + bc$



Note: edges may include query, DML, DDL, DCL; order of operations is also evident