The (Potential) Perks of Integrating Provenance Support into Database Engines

Boris Glavic, Gopal Shankar[1]

Illinois Institute of Technology
DBGroup

GCASR 2013, May 3, 2013

[1] Master student

Outline

Given a piece of data

- How do we know ...
  - which data it is derived from?
  - which transformations (SQL) where used to create it?
  - who created it?
  - ...

Example

result

	shop	rev
t_1	Migros	125
t_2	Coop	25

Given a piece of data

- How do we know ...
  - which data it is derived from?
  - which transformations (SQL) where used to create it?
  - who created it?
  - ...

Example

Compute the revenue for each shop as sum of prices of items sold

Example

result

	shop	rev
t_1	Migros	125
t_2	Coop	25

↑

```
SELECT shop,
    sum(price) AS rev
FROM sales, items
WHERE itemId = id
GROUP BY shop
```

↑ ↑

sales

	shop	itemId
s_1	Migros	1
s_2	Migros	3
s_3	Coop	3

items

	id	price
i_1	1	100
i_2	2	10
i_3	3	25

Provenance in Databases

ILLINOIS INSTITUTE OF TECHNOLOGY

Given a piece of data

- How do we know ...
  - which data it is derived from?
  - which transformations (SQL) where used to create it?
  - who created it?
  - ...

Definition (Data Provenance)

Information about the origin and creation process of data.

Example

result

	shop	rev
t_1	Migros	125
t_2	Coop	25

↑

```
SELECT shop,
    sum(price) AS rev
FROM sales, items
WHERE itemId = id
GROUP BY shop
```

↑ ↑

sales

	shop	itemId
s_1	Migros	1
s_2	Migros	3
s_3	Coop	3

items

	id	price
i_1	1	100
i_2	2	10
i_3	3	25

Provenance in Databases

ILLINOIS INSTITUTE OF TECHNOLOGY

Given a piece of data

- How do we know ...
  - which data it is derived from?
  - which transformations (SQL) where used to create it?
  - who created it?
  - ...

Definition (In this talk)

Data-Dependencies: which input data does query output depend on.

Example

result

	shop	rev
t_1	Migros	125
t_2	Coop	25

↑

```
SELECT shop,
    sum(price) AS rev
FROM sales, items
WHERE itemId = id
GROUP BY shop
```

↑ ↑

sales

	shop	itemId
s_1	Migros	1
s_2	Migros	3
s_3	Coop	3

items

	id	price
i_1	1	100
i_2	2	10
i_3	3	25

Provenance Usage

ILLINOIS INSTITUTE OF TECHNOLOGY

Use Cases

- Debugging (tracking the sources of errors)
- Propagating annotations
- Gain deeper understanding of data and transformations
  - Estimate quality, trust
- Access Control (PBAC)
- Supporting technology (e.g., Probabilistic databases, Creating test databases)

Application Domains

- Complex database queries, e.g., datawarehousing
- E-science and curated databases
- Data integration

How to compute Provenance?

ILLINOIS INSTITUTE OF TECHNOLOGY

Annotation Propagation

- Annotate inputs
- Propagate these annotations during query processing
    - Combine/filter annotations based on data-dependencies of operators

Example

- Annote inputs with tuple IDs (singleton set)
- Join: Set union
- Duplicate Removal: Set union

Example

Result

	a
t	1

```
SELECT DISTINCT R.a
FROM R, S
WHERE R.a = S.c;
```

R

	a	b
r_1	1	2
r_2	3	6

S

	c	d
s_1	1	2
s_2	1	3

Example

- Annote inputs with tuple IDs (singleton set)
- Join: Set union
- Duplicate Removal: Set union

$$(\{r_1\} \cup \{s_1\}) \cup (\{r_1\} \cup \{s_2\})$$
$$= \{r_1, s_1, s_2\}$$

Example

Result

	a
$\{r_1, s_1, s_2\}$	1

↑

```sql
SELECT DISTINCT R.a
FROM R, S
WHERE R.a = S.c;
```

↑ ↑

R

	a	b
r_1	1	2
r_2	3	6

S

	c	d
s_1	1	2
s_2	1	3

Relational encoding of provenance

1. Standard relations
   - Extend query result tuples with complete input tuples in provenance (or only ID columns)
   - Duplicate tuples to fit in provenance
2. New data types (DBMS extensibility mechanism)
   - Use new/existing complex data type to represent provenance

Example (Provenance)

$$\{< r_1, s_1 >, < r_1, s_2 >\}$$

Example

Result

	a
t	1

```
SELECT DISTINCT R.a
FROM R, S
WHERE R.a = S.c;
```

R

	a	b
r_1	1	2
r_2	3	6

S

	c	d
s_1	1	2
s_2	1	3

Relation with Provenance + Query Result Data

- **Data**: result tuple + all tuples from input tables in provenance
  - Result tuple might have to be duplicated!
- **Schema**: + input attributes (renamed)

Example (Query Result)

R

a	b
1	2
3	4

S

c	d
1	2
1	3

$\xrightarrow{q}$

Result q

a
1

Representation - Standard Relations

ILLINOIS INSTITUTE OF TECHNOLOGY

Relation with Provenance + Query Result Data

- **Data**: result tuple + all tuples from input tables in provenance
  - Result tuple might have to be duplicated!
- **Schema**: + input attributes (renamed)

Example (Provenance Representation)

R

a	b
1	2
3	4

S

c	d
1	2
1	3

$\xrightarrow{q}$

Result + Provenance

a	P(a)	P(b)	P(c)	P(d)
1	1	2	1	2
1	1	2	1	3

Representation - Standard Relations

Relation with Provenance + Query Result Data

- **Data**: result tuple + all tuples from input tables in provenance
  - Result tuple might have to be duplicated!
- **Schema**: + input attributes (renamed)

Example (Provenance Representation)

R

a	b
1	2
3	4

S

c	d
1	2
1	3

$\xrightarrow{q}$

Result + Provenance

a	P(a)	P(b)	P(c)	P(d)
1	1	2	1	2
1	1	2	1	3

Representation - Single Value

Relation with Query Result Data + Extra Attribute

- **Data**: result tuple + new attribute to store provenance
  - Value of new attribute: complete provenance for this output
- **Schema**: + new provenance attribute

Example (Query Result)

R

a	b
1	2
3	4

S

c	d
1	2
1	3

$\xrightarrow{q}$

Result q

a
1

Representation - Single Value

Relation with Query Result Data + Extra Attribute

- **Data**: result tuple + new attribute to store provenance
  - Value of new attribute: complete provenance for this output
- **Schema**: + new provenance attribute

Example (Provenance Representation)

R

a	b
1	2
3	4

S

c	d
1	2
1	3

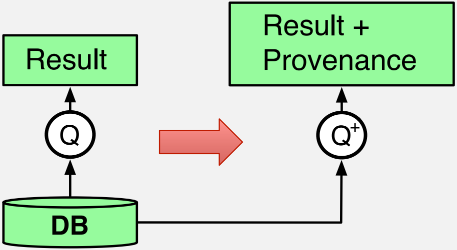
$\xrightarrow{q}$

Result + Provenance

a	Prov
1	$\{r_1, s_1, s_2\}$

Query rewrite - Standard relations

- Take original query q and rewrite into q^+:
  - Attach provenance to inputs (on-the-fly or join existing)
  - Propagate trough operations
  - $\Rightarrow$Compute original results $+$ provenance



State-of-the-art - Provenance Generation

Advantages

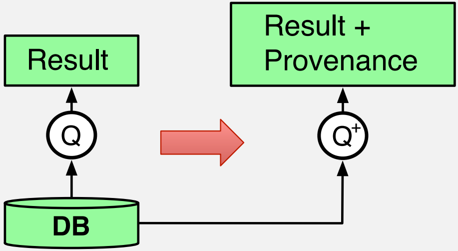
- Reuse of existing DB technology
- DBMS independence (to some extend)

Disadvantages

- Unnecessarily complicated and inefficient computation
  - Impossible to propagate provenance through some operators
    - ⇒need to duplicate parts of the query
- Hard to benefit from overlap and structure in provenance
- Complex queries that are hard to optimize

Query rewrite - Single Value

- Take original query q and rewrite into q^+:
  - Attach provenance to inputs (on-the-fly or join existing)
  - User-defined functions (UDF) to combine propagated provenance



State-of-the-art - Provenance Generation

Advantages

- Reuse of existing DB technology
- DBMS independence (to some extend)

Disadvantages

- Unnecessarily complicated and inefficient computation
  - Impossible to propagate provenance through some operators
  - DBMS not optimized for very large attribute values
  - Optimizer has to handle provenance computations as black-boxes
- Hard to benefit from overlap and structure in provenance

Fundamental Problem - Performance

Implications of loose integration of provenance

- Severe performance limitations
  - SQL and DBMS not designed for provenance ⇒complex queries
  - Different data-flow (mostly append only, less reduction) ⇒relational execution engine not a good fit
  - Provenance structure predictable and overlapping ⇒no exploited
- ⇒hinder broad adaptation
  - Access Control (performance critical)
  - Supporting technology, e.g., probabilistic data (performance critical)

What can be gained by full integration of provenance in DBMS engine?

- Provenance-aware operators
- Provenance optimized data-flow during query processing
- Performance optimized compression techniques

Outline

Research Approach

Tight Integration of Provenance Computation in the DBMS core

- Provenance-aware operators
  - Simplify queries ⇒easier to optimize
  - Direct propagation without "rerouting" ⇒less redundancy
- Compression schemes exploiting provenance structure
  - Faster compression/decompression ⇒applicable during provenance computation
  - Execute operations on compressed provenance directly
  - Approximate provenance
- Investigate non-traditional data-flows for provenance
  - Cache provenance between queries
  - Share provenance between operators ⇒divert from traditional operator base processing
  - ...

Research Approach

ILLINOIS INSTITUTE OF TECHNOLOGY

Tight Integration of Provenance Computation in the DBMS core

- Provenance-aware operators
  - Simplify queries ⇒easier to optimize
  - Direct propagation without "rerouting" ⇒less redundancy
- Compression schemes exploiting provenance structure
  - Faster compression/decompression ⇒applicable during provenance computation
  - Execute operations on compressed provenance directly
  - Approximate provenance
- Investigate non-traditional data-flows for provenance
  - Cache provenance between queries
  - Share provenance between operators ⇒divert from traditional operator base processing
  - ...

Declarative Summaries

Rational

- Overlap in provenance
- Avoid generating large provenance for intermediates if not used in final provenance
  - Performance gain!

Approach

- Use declarative descriptions as placeholders for actual provenance
- Only expand when necessary
- Suited to present provenance to user?

Declarative Summaries

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

```
SELECT count(*), R.b
  (SELECT R.a
  FROM R
  WHERE EXISTS (SELECT * FROM S WHERE S.c > 5) AND R.a < 10)
GROUP BY R.b
HAVING count(*) > 1000
```

- Each output will have all tuples from S with $S.c > 5$ in its provenance
- Propagation will attach this potentially large set of tuples
  - to every tuple from R
  - many of these tuples may belong to groups that will be filtered eventually
  - $\Rightarrow$unnecessary work

Example

```
SELECT count(*), R.b
  (SELECT R.a
  FROM R
  WHERE EXISTS (SELECT * FROM S WHERE S.c > 5) AND R.a < 10)
GROUP BY R.b
HAVING count(*) > 1000
```

Provenance

a	Prov
1	$\{r_1, s_1, s_3, s_5, \ldots\}$
13	$\{r_5, s_1, s_3, s_5, \ldots\}$
...	...

Using Declarative Summarization

a	Prov
1	$\{r_1, \sigma_{c>5}(S)\}$
13	$\{r_5, \sigma_{c>5}(S)\}$
...	...

Non-traditional data-flows

Outline

1. Motivation
2. Research Directions
3. **Improving Performance with Provenance-aware Operators**
4. Conclusions

Problem - Performance

Data-flow and access patterns

- Provenance computation $\neq$ Regular queries
  - Tuple breadth constantly increasing
  - Coupling of input data with intermediate data
  - Overlap

Direct propagation through operators not always possible

- Adding additional rows to input of an aggregation
- $\Rightarrow$change result!

Provenance-aware Operators

Approach

- Operators optimized for provenance specific tasks
- Support direct propagation of provenance
- Implement and evaluate new operators in **Perm**
  - Perm is a relational provenance system based on query rewrite
  - Implemented as an extension to Postgres

Rationale

- Less complex queries
- Specialized code
- Less redundancy in computation

Case study: AggProject - A Provenance-aware Aggregation Operator

Problem

- Cannot propagate provenance through aggregation directly
  - Would change aggregation results
- Compute aggregation provenance
  - Compute the original aggregation
  - Compute provenance for query below aggregation
  - Join these two
- Complexity
  - Each level of aggregation
    - Doubles query size
    - Adds additional join
  - $\Rightarrow$Exponential in levels of aggregations!

Solution

- Develop a new type of aggregation that caches input provenance and attaches it to aggregation results

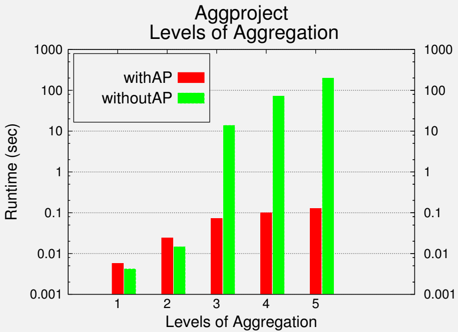
Implementation in Perm

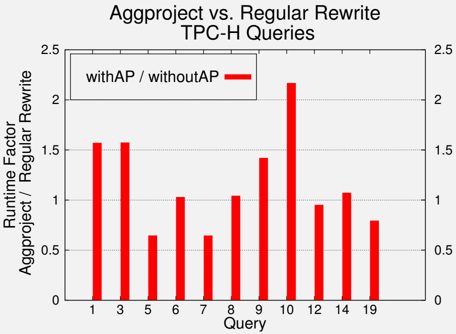
- Add AGGPROJECT as new clause to SQL
- Physical aggproject operator using aggregation by sorting
- "Hack" integration with query optimizer
- Integration with provenance query rewrites
- Support multiple levels of aggregation

ILLINOIS INSTITUTE OF TECHNOLOGY

Setup

- TPC-H Benchmark Dataset (1GB)
- Perm vs. Perm with Aggproject





Next Steps

AggProject

- Develop cost-model
- Implemented Hash-based variant
  - In-memory
  - With disk cashing

Provenance-aware Operators

- Set operations
- Nested subqueries

Outline

- Provenance become more and more important
- State-of-the-art of relational provenance systems unsatisfactory
  - Query rewrite, custom data types, stored procedures
  - Seriously performance limitations
    - Reduces adaptation for use cases that benefit from provenance
- Integration of provenance in DBMS engine core
  - Potential for addressing the performance bottleneck

Future Work

ILLINOIS INSTITUTE OF TECHNOLOGY

- Continue provenance-aware operators approach
- Efficient provenance representations and declarative summarizations
  - Develop model and equivalences
  - Cost model and performance analysis
- Non-traditional data flows
  - Caching provenance
  - Operators shared provenance data structures
  - Indexing

Questions?

- **Homepage**: `http://www.cs.iit.edu/~glavic/`
- **DBGroup**: `http://www.cs.iit.edu/~dbgroup/`
- **Perm**:
  - Project page: `http://www.cs.iit.edu/~dbgroup/research/perm.php`
  - Svn repository: `https://permdbms.svn.sourceforge.net/svnroot/permdbms/trunk`