Aggregation and Ordering in Factorized Databases

Bakibayev, Kočiský, Olteanu, and Závodný
University of Oxford

VLDB Sept 2, 2014

http://www.cs.ox.ac.uk/projects/FDB/
Outline

What are Factorized Databases?

Applications

A Glimpse at Aggregating Factorized Data
**Orders**

<table>
<thead>
<tr>
<th>customer</th>
<th>day</th>
<th>pizza</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mario</td>
<td>Monday</td>
<td>Capricciosa</td>
</tr>
<tr>
<td>Mario</td>
<td>Friday</td>
<td>Capricciosa</td>
</tr>
<tr>
<td>Pietro</td>
<td>Friday</td>
<td>Hawaii</td>
</tr>
<tr>
<td>Lucia</td>
<td>Friday</td>
<td>Hawaii</td>
</tr>
</tbody>
</table>

**Pizzas**

<table>
<thead>
<tr>
<th>pizza</th>
<th>item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capricciosa</td>
<td>base</td>
</tr>
<tr>
<td>Capricciosa</td>
<td>ham</td>
</tr>
<tr>
<td>Capricciosa</td>
<td>mushrooms</td>
</tr>
<tr>
<td>Hawaii</td>
<td>base</td>
</tr>
<tr>
<td>Hawaii</td>
<td>ham</td>
</tr>
<tr>
<td>Hawaii</td>
<td>pineapple</td>
</tr>
</tbody>
</table>

**Items**

<table>
<thead>
<tr>
<th>item</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>6</td>
</tr>
<tr>
<td>ham</td>
<td>1</td>
</tr>
<tr>
<td>mushrooms</td>
<td>1</td>
</tr>
<tr>
<td>pineapple</td>
<td>2</td>
</tr>
</tbody>
</table>

Consider the natural join of the three relations above:

**Orders ⨟ Pizzas ⨟ Items**

<table>
<thead>
<tr>
<th>customer</th>
<th>day</th>
<th>pizza</th>
<th>item</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mario</td>
<td>Monday</td>
<td>Capricciosa</td>
<td>base</td>
<td>6</td>
</tr>
<tr>
<td>Mario</td>
<td>Monday</td>
<td>Capricciosa</td>
<td>ham</td>
<td>1</td>
</tr>
<tr>
<td>Mario</td>
<td>Monday</td>
<td>Capricciosa</td>
<td>mushrooms</td>
<td>1</td>
</tr>
<tr>
<td>Mario</td>
<td>Friday</td>
<td>Capricciosa</td>
<td>base</td>
<td>6</td>
</tr>
<tr>
<td>Mario</td>
<td>Friday</td>
<td>Capricciosa</td>
<td>ham</td>
<td>1</td>
</tr>
<tr>
<td>Mario</td>
<td>Friday</td>
<td>Capricciosa</td>
<td>mushrooms</td>
<td>1</td>
</tr>
<tr>
<td>. . . . .</td>
<td>. . . .</td>
<td>. . . .</td>
<td>. . . .</td>
<td>. . . .</td>
</tr>
</tbody>
</table>
A *flat* relational algebra expression encoding the above query result is:

\[
\langle \text{Mario} \rangle \times \langle \text{Monday} \rangle \times \langle \text{Capricciosa} \rangle \times \langle \text{base} \rangle \times \langle 6 \rangle \cup \\
\langle \text{Mario} \rangle \times \langle \text{Monday} \rangle \times \langle \text{Capricciosa} \rangle \times \langle \text{ham} \rangle \times \langle 1 \rangle \cup \\
\langle \text{Mario} \rangle \times \langle \text{Monday} \rangle \times \langle \text{Capricciosa} \rangle \times \langle \text{mushrooms} \rangle \times \langle 1 \rangle \cup \\
\langle \text{Mario} \rangle \times \langle \text{Friday} \rangle \times \langle \text{Capricciosa} \rangle \times \langle \text{base} \rangle \times \langle 6 \rangle \cup \\
\langle \text{Mario} \rangle \times \langle \text{Friday} \rangle \times \langle \text{Capricciosa} \rangle \times \langle \text{ham} \rangle \times \langle 1 \rangle \cup \\
\langle \text{Mario} \rangle \times \langle \text{Friday} \rangle \times \langle \text{Capricciosa} \rangle \times \langle \text{mushrooms} \rangle \times \langle 1 \rangle \cup \ldots
\]

It uses relational product (\(\times\)), union (\(\cup\)), and singleton relations (e.g., \(\langle 1 \rangle\)).

- The attribute names are not shown to avoid clutter.
The previous relational expression entails lots of redundancy due to the joins:

\[
\langle \text{Mario} \rangle \times \langle \text{Monday} \rangle \times \langle \text{Capricciosa} \rangle \times \langle \text{base} \rangle \times \langle 6 \rangle \cup \\
\langle \text{Mario} \rangle \times \langle \text{Monday} \rangle \times \langle \text{Capricciosa} \rangle \times \langle \text{ham} \rangle \times \langle 1 \rangle \cup \\
\langle \text{Mario} \rangle \times \langle \text{Monday} \rangle \times \langle \text{Capricciosa} \rangle \times \langle \text{mushrooms} \rangle \times \langle 1 \rangle \cup \\
\langle \text{Mario} \rangle \times \langle \text{Friday} \rangle \times \langle \text{Capricciosa} \rangle \times \langle \text{base} \rangle \times \langle 6 \rangle \cup \\
\langle \text{Mario} \rangle \times \langle \text{Friday} \rangle \times \langle \text{Capricciosa} \rangle \times \langle \text{ham} \rangle \times \langle 1 \rangle \cup \\
\langle \text{Mario} \rangle \times \langle \text{Friday} \rangle \times \langle \text{Capricciosa} \rangle \times \langle \text{mushrooms} \rangle \times \langle 1 \rangle \cup \ldots
\]

We can factorize the expression following the join structure, e.g.:

\[
\langle \text{Capricciosa} \rangle \times ((\langle \text{Monday} \rangle \times \langle \text{Mario} \rangle \cup \langle \text{Friday} \rangle \times \langle \text{Mario} \rangle) \\
\times (\langle \text{base} \rangle \times \langle 6 \rangle \cup \langle \text{ham} \rangle \times \langle 1 \rangle \cup \langle \text{mushrooms} \rangle \times \langle 1 \rangle) \\
\cup \langle \text{Hawaii} \rangle \times \langle \text{Friday} \rangle \times ((\langle \text{Lucia} \rangle \cup \langle \text{Pietro} \rangle) \\
\times (\langle \text{base} \rangle \times \langle 6 \rangle \cup \langle \text{ham} \rangle \times \langle 1 \rangle \cup \langle \text{pineapple} \rangle \times \langle 2 \rangle))
\]

There are several algebraically equivalent factorized representations defined by distributivity of product over union and commutativity of product and union.
Properties of Factorized Representations

Factorized representations of results of queries with select, project, join, aggregate, groupby, and orderby operators:

- **Very high compression rate**
  - Can be exponentially more succinct than the relations they encode.
  - Arbitrarily better than generic compression schemes, e.g., bzip2
  - Factorized representations of asymptotically-tight size bounds computable directly from input database and query

- **Querying in the compressed domain**
  - Factorizations are relational expressions
  - We developed the FDB in-memory query engine for this purpose

- **Constant-delay enumeration of represented tuples**
  - Tuple iteration as fast as listing them from equivalent flat relations
Outline

What are Factorized Databases?

Applications

A Glimpse at Aggregating Factorized Data
Excerpt from *F1: A Distributed SQL Database That Scales*. PVLDB’13.

- Google’s DB supporting their lucrative AdWords business
- Database factorization increases data locality for common access patterns
  - Tables pre-joined using a nesting structure defined by key-fkey constraints
- Data partitioned across servers into factorization fragments
Spot the Factorized Database!

Figure 3: (a) In relational domains, design matrices $X$ have large blocks of repeating patterns (example from Figure 2). (b) Repeating patterns in $X$ can be formalized by a block notation (see section 2.3) which stems directly from the relational structure of the original data. Machine learning methods have to make use of repeating patterns in $X$ to scale to large relational datasets.

Excerpt from *Scaling Factorization Machines to Relational Data*. PVLDB’13.

- Feature vectors for predictive modelling represented as very large design matrices (i.e. relations with high cardinality)
- Standard learning algorithms cannot scale on design matrix representation
- Use repeating patterns in the design matrix as key to scalability
Fig. 1. Two completed survey forms and a world-set relation representing the possible worlds with unique social security numbers.

Excerpt from 10^{10^6} Worlds and Beyond: Efficient Representation and Processing of Incomplete Information. ICDE'07.

Managing a large set of possibilities or choices:

- Configuration problems (space of valid solutions)
- Incomplete information (space of possible worlds)
An important class of propositional formulas that play a special role in probabilistic databases are read-once formulas. We restrict our discussion to the case when all random variables $X$ are Boolean variables.

$\Phi_1$ is called read-once if there is a formula $\Phi_1'$ equivalent to $\Phi_1$ such that every variable occurs at most once in $\Phi_1'$. For example:

$$\Phi_1 = X_1Y_1 \lor X_1Y_2 \lor X_2Y_3 \lor X_2Y_4 \lor X_2Y_5$$

is read-once because it is equivalent to the following formula:

$$\Phi_1' = X_1(Y_1 \lor Y_2) \lor X_2(Y_3 \lor Y_4 \lor Y_5)$$


Provenance and probabilistic data:

- Compact encoding for large provenance
- Factorization of provenance is used for efficient query evaluation in probabilistic databases
Outline

What are Factorized Databases?

Applications

A Glimpse at Aggregating Factorized Data
Aggregating Factorized Data

We only present here COUNT and SUM aggregation functions.

COUNT($F$) is the number of tuples in a factorization $F$:

- $\text{COUNT}(\langle a \rangle) = 1$.
- $\text{COUNT}(F_1 \cup \cdots \cup F_k) = \text{COUNT}(F_1) + \cdots + \text{COUNT}(F_k)$.
- $\text{COUNT}(F_1 \times \cdots \times F_k) = \text{COUNT}(F_1) \cdot \cdots \cdot \text{COUNT}(F_k)$.

SUM$_A(F)$ is the sum of all values of attribute $A$ in a factorization $F$:

- $\text{SUM}_A(\langle a \rangle) = a$, if the singleton $\langle a \rangle$ has attribute $A$.
- $\text{SUM}_A(F_1 \cup \cdots \cup F_k) = \text{SUM}_A(F_1) + \cdots + \text{SUM}_A(F_k)$.
- $\text{SUM}_A(F_1 \times \cdots \times F_k) = \text{SUM}_A(F_1) \cdot \text{COUNT}(F_2) \cdot \cdots \cdot \text{COUNT}(F_k)$, where wlog values for attribute $A$ are in expression $F_1$. 

13 / 19
Aggregation by Example

- Recall the natural join of Orders, Pizzas, and Items
- We would like to find the overall sales per customer
- Assume the factorization structure discussed before (leftmost below)

Example of possible evaluation plans:

1. First restructure for GROUP-BY, then aggregate

   - First restructure for GROUP-BY, then aggregate

2. Intertwine restructuring for GROUP-BY and partial aggregation

   - Intertwine restructuring for GROUP-BY and partial aggregation
Query Evaluation Step by Step

Let us consider the second evaluation plan:

![Diagram of query evaluation steps]

The initial factorization with the structure highlighted above:

\[
\langle Capricciosa \rangle \times (\langle Monday \rangle \times \langle Mario \rangle \cup \langle Friday \rangle \times \langle Mario \rangle) \\
\times (\langle base \rangle \times \langle 6 \rangle \cup \langle ham \rangle \times \langle 1 \rangle \cup \langle mushrooms \rangle \times \langle 1 \rangle) \\
\cup \langle Hawaii \rangle \times \langle Friday \rangle \times (\langle Lucia \rangle \cup \langle Pietro \rangle) \\
\times (\langle base \rangle \times \langle 6 \rangle \cup \langle ham \rangle \times \langle 1 \rangle \cup \langle pineapple \rangle \times \langle 2 \rangle) 
\]
Let us consider the second evaluation plan:

The factorization after partial aggregation with the structure highlighted above:

\[
\langle \text{Capricciosa} \rangle \times (\langle \text{Monday} \rangle \times \langle \text{Mario} \rangle \cup \langle \text{Friday} \rangle \times \langle \text{Mario} \rangle) \\
\times \langle 8 \rangle \\
\cup \langle \text{Hawaii} \rangle \times \langle \text{Friday} \rangle \times (\langle \text{Lucia} \rangle \cup \langle \text{Pietro} \rangle) \\
\times \langle 9 \rangle
\]
Let us consider the second evaluation plan:

The factorization after restructuring with the structure highlighted above:

\[
\langle Lucia \rangle \times \langle Hawaii \rangle \times \langle Friday \rangle \times \langle 9 \rangle \cup \\
\langle Mario \rangle \times \langle Capricciosa \rangle \times (\langle Monday \rangle \cup \langle Friday \rangle) \times \langle 8 \rangle \cup \\
\langle Pietro \rangle \times \langle Hawaii \rangle \times \langle Friday \rangle \times \langle 9 \rangle
\]
Let us consider the second evaluation plan:

The factorization after final aggregation with the structure highlighted above:

\[
\langle Lucia \rangle \times \langle 9 \rangle \cup \\
\langle Mario \rangle \times \langle 16 \rangle \cup \\
\langle Pietro \rangle \times \langle 9 \rangle
\]
Thank you!