Factorized databases II

Jakub Zavodny (University of Oxford, UK)

DATA ANALYSIS AND MODELING LAB
Palacky University, Olomouc, Czech Republic

INVESTMENTS IN EDUCATION DEVELOPMENT

Factorised Databases 2.

Factorised Database Engine

Jakub Závodný, University of Oxford

based on joint work with PhD supervisor Dan Olteanu

Factorised Relations

$$\Big(\langle \mathit{Tesco}\rangle \times (\langle \mathit{food}\rangle \cup \langle \mathit{clothes}\rangle) \cup \langle \mathit{Lidl}\rangle \times \langle \mathit{food}\rangle\Big) \times (\langle \mathit{CzechRep}\rangle \cup \langle \mathit{Slovakia}\rangle) \cup$$
$$(\langle \mathit{Tesco}\rangle \cup \langle \mathit{Lidl}\rangle \cup \langle \mathit{Aldi}\rangle) \times \langle \mathit{food}\rangle \times (\langle \mathit{Poland}\rangle \cup \langle \mathit{UK}\rangle) \cup$$
$$(\langle \mathit{Tesco}\rangle \cup \langle \mathit{Aldi}\rangle) \times \langle \mathit{books}\rangle \times (\langle \mathit{UK}\rangle \cup \langle \mathit{Spain}\rangle)$$

Representations of relations that

- compactly encode combinations of groups of values.
- allow for enumerating tuples with constant delay.
- allow query evaluation *without expanding* to flat relations
  - Using factorisations can boost query evaluation (processing a smaller representation).

Key Results

1. How compact can factorised query results be?

- Asymptotic size bounds for factorisations of query results.
- Parameter $s(Q)$ characterising the factorisability of results of Q.

2. Can such factorisations speed up query evaluation?

- Compute factorisations directly from query and input data.
- Optimisation and evaluation techniques for queries with selections, projections, joins, aggregates and ordering on factorisations.
- FDB: a main-memory engine using factorised relations.

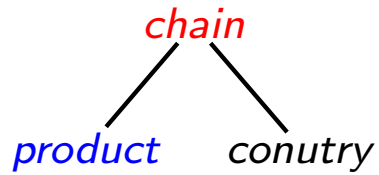
Key Challenges

2. Can such factorisations speed up query evaluation?

Factorisation Trees

Uniform nesting structures $\longleftrightarrow$ defined by *factorisation trees* (f-trees).

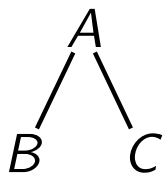
$$\bigcup_{Chains} \Big(\langle chain \rangle \times \Big(\bigcup_{Products} \langle product \rangle \Big) \times \Big(\bigcup_{Countries} \langle country \rangle \Big) \Big) \longleftrightarrow$$



$$\bigcup_{Chains} \Big(\langle chain \rangle \times \Big(\bigcup_{Products} \langle product \rangle \times \Big(\bigcup_{Countries} \langle country \rangle \Big) \Big) \Big) \longleftrightarrow$$

Factorisation Trees

F-tree **defines** the factorisation of a relation.



$+$

A	B	C
x	1	2
x	1	3
x	2	2
x	2	3
y	4	4

$$\Downarrow$$

$$\begin{aligned} &\langle x \rangle \times (\langle 1 \rangle \cup \langle 2 \rangle) \times (\langle 2 \rangle \cup \langle 3 \rangle) \\ \cup\, &\langle y \rangle \times \langle 4 \rangle \qquad\qquad \times \langle 4 \rangle \end{aligned}$$

Example: Evaluating a Selection on a Factorised Relation

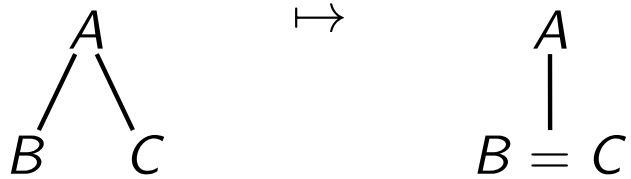
Evaluate $\sigma_{B=C} R(A, B, C)$:

A	B	C
x	1	2
x	1	3
x	2	2
x	2	3
y	4	4

$\mapsto$

A	B = C
x	2
y	4

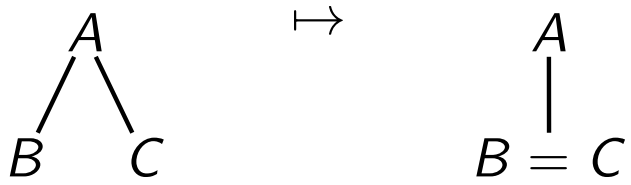
$$\langle x \rangle \times (\langle 1 \rangle \cup \langle 2 \rangle) \times (\langle 2 \rangle \cup \langle 3 \rangle) \cup \langle y \rangle \times \langle 4 \rangle \quad \mapsto \quad \langle x \rangle \times \langle 2 \rangle \cup \langle y \rangle \times \langle 4 \rangle$$



Example: Evaluating a Selection on a Factorised Relation

To evaluate $\sigma_{B=C}$:

- **Merge nodes** B, C in the tree



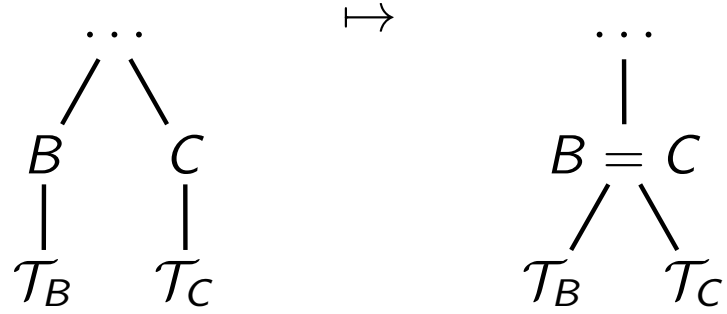
- **Merge-join corresponding unions** over B and C in linear time.

$$\begin{array}{lll} \langle x\rangle \times (\langle 1\rangle \cup \langle 2\rangle) \times (\langle 2\rangle \cup \langle 3\rangle) \cup & \mapsto & \langle x\rangle \times \langle 2\rangle \cup \\ \langle y\rangle \times \langle 4\rangle \qquad\quad \times \langle 4\rangle & & \langle y\rangle \times \langle 4\rangle \end{array}$$

Query Evaluation: Merge Selection

In general, we can evaluate $\sigma_{B=C}$ if B and C are **siblings**:

- **Merge nodes** B, C in the tree



- **Merge-join corresponding unions** over B and C in linear time.

$$\ldots(\langle b_1\rangle \times B_1 \cup \langle b_2\rangle \times B_2 \cup \cdots \cup \langle b_n\rangle \times B_n) \times (\langle c_1\rangle \times C_1 \cup \langle c_2\rangle \times C_2 \cup \cdots \cup \langle c_m\rangle \times C_m)\ldots$$

$$\mapsto$$

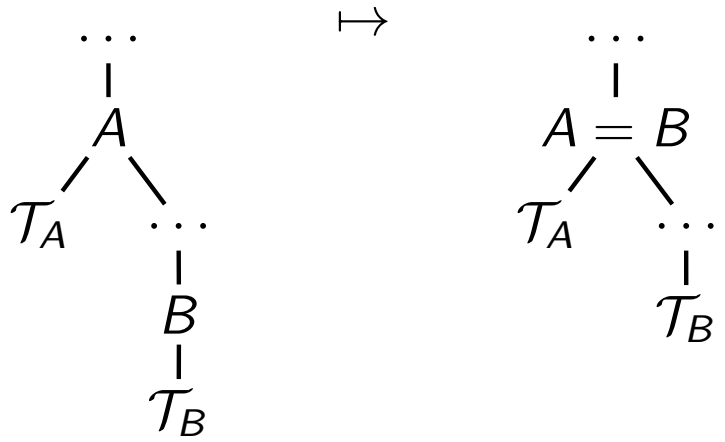
$$\ldots(\langle b_{i_1} = c_{j_1}\rangle \times B_{i_1} \times C_{j_1} \cup \cdots \cup \langle b_{i_1} = c_{j_1}\rangle \times B_{i_k} \times C_{j_k})\ldots$$

Call this the **merge operator**.

Query Evaluation: Absorb Selection

Also, we can evaluate $\sigma_{A=B}$ if A is **ancestor** of B:

- **Absorb node** B into A in the tree



- **Filter unions** over B in linear time.

$$\ldots\langle a\rangle \times \left(\bigcup \ldots (\langle b_1\rangle \times B_1 \cup \langle b_2\rangle \times B_2 \cup \cdots \cup \langle b_n\rangle \times B_n) \ldots\right)$$

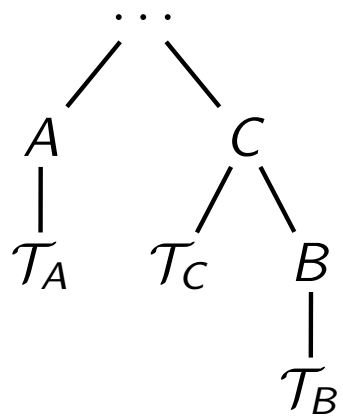
$$\mapsto$$

$$\ldots\langle a\rangle \times \left(\bigcup \ldots (\langle b_i = a\rangle \times B_i) \ldots\right) \quad \mapsto \quad \langle a\rangle \times \left(\bigcup \ldots B_i \ldots\right)$$

Call this the **absorb operator**.

Query Evaluation: Selection

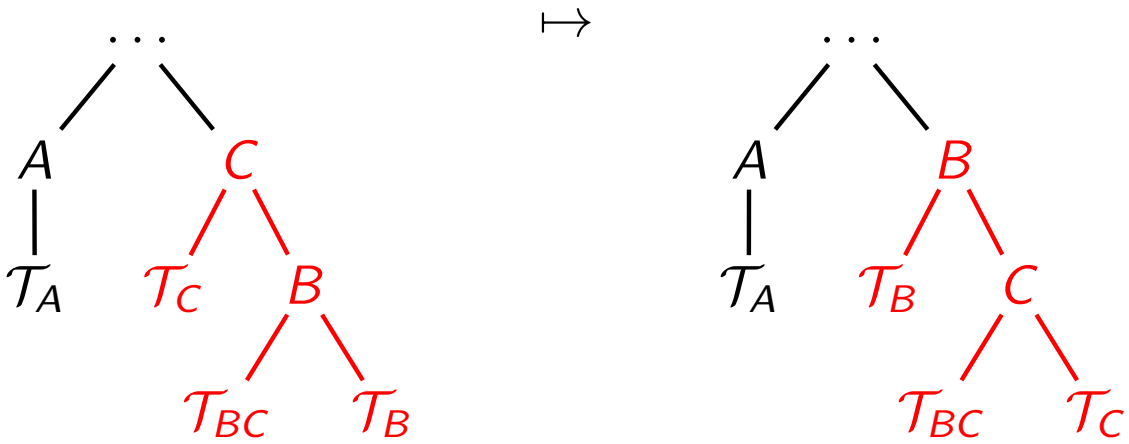
How to join $A = B$ when A and B are not siblings or descendants?



Query Evaluation: Restructuring Operators

How to join $A = B$ when A and B are not siblings / descendants?

Restructure the factorisation.



Restructuring operators

- **Swap with parent** (re-group terms)
- **Push up** (factor out common expressions)

in quasilinear time in $(|\text{input}| + |\text{output}|)$.

Query Evaluation

Operators on factorisations **defined by** operators on f-trees:

Equality selection operators

- **Merge** siblings
- **Absorb** descendant

Restructuring operators

- **Swap** with parent
- **Push up**

+

- **Project** away a leaf attribute
- **Cartesian product** = simple tree concatenation, constant time!

Any SPJ query can be evaluated by a sequence of these operators
(called an **f-plan** for the query).

Query Evaluation: Example

PlaysFor

player	team
Messi	Barcelona
Villa	Barcelona
Cech	Chelsea
Torres	Chelsea
van Persie	Arsenal

CompetesIn

team	league
Barcelona	Primera
Barcelona	Champions
Chelsea	Premier
Chelsea	Champions
Arsenal	Premier

LeagueStadium

league	stadium
Primera	CampNou
Champions	CampNou
Champions	Wembley
Premier	Stamford
Premier	Wembley

BasedIn

team	city
Barcelona	Barcelona
Chelsea	London
Arsenal	London

IsIn

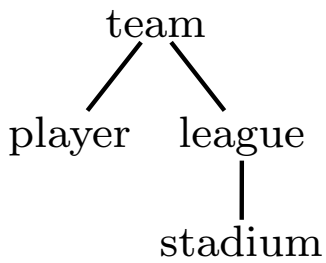
stadium	city
CampNou	Barcelona
Wembley	London
Stamford	London

Which players can I see on which stadiums?
Which stadiums and which teams are in the same city?

Query Evaluation: Example

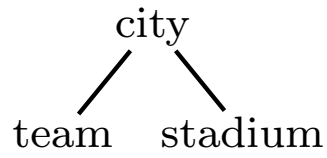
Which players can I see on which stadiums?
Which stadiums and which teams are in the same city?

$$\text{PlaysFor} \bowtie_{\text{team}} \text{CompetesIn} \bowtie_{\text{league}} \text{LeagueStadium}$$



$$\begin{aligned}
&\Big(\langle \mathit{FCBarcelona}\rangle \times (\langle \mathit{Messi}\rangle \cup \langle \mathit{Villa}\rangle) \times \\
&\qquad \times \big(\langle \mathit{Primera}\rangle \times \langle \mathit{CampNou}\rangle \cup \\
&\qquad\quad \langle \mathit{Champions}\rangle \times (\langle \mathit{CampNou}\rangle \cup \langle \mathit{Wembley}\rangle)\big) \cup \\
&\langle \mathit{Chelsea}\rangle \times (\langle \mathit{Cech}\rangle \cup \langle \mathit{Torres}\rangle) \times \\
&\qquad \times \big(\langle \mathit{Premier}\rangle \times (\langle \mathit{Stamford}\rangle \cup \langle \mathit{Wembley}\rangle)\big) \cup \\
&\qquad\quad \langle \mathit{Champions}\rangle \times (\langle \mathit{CampNou}\rangle \cup \langle \mathit{Wembley}\rangle)\big) \cup \\
&\langle \mathit{Arsenal}\rangle \times \langle \mathit{van\ Persie}\rangle \times \\
&\qquad \times \big(\langle \mathit{Premier}\rangle \times (\langle \mathit{Stamford}\rangle \cup \langle \mathit{Wembley}\rangle)\big)\Big)
\end{aligned}$$

$$\text{BasedIn} \bowtie_{\text{city}} \text{IsIn}$$



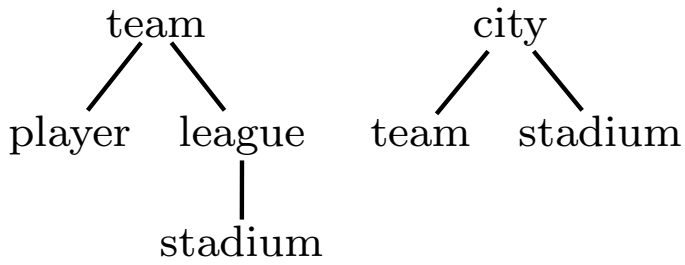
$$\begin{aligned}
&\Big(\langle \mathit{Barcelona}\rangle \times \langle \mathit{FCBarcelona}\rangle \\
&\qquad \times \langle \mathit{CampNou}\rangle \cup \\
&\langle \mathit{London}\rangle \times (\langle \mathit{Chelsea}\rangle \cup \langle \mathit{Arsenal}\rangle) \\
&\qquad \times (\langle \mathit{Wembley}\rangle \cup \langle \mathit{Stamford}\rangle)\Big)
\end{aligned}$$

Which players *from the same city* can I see on which stadiums?

Query Evaluation: Example

Which players *from the same city* can I see on which stadiums?

$$(\text{PlaysFor} \bowtie_{\text{team}} \text{CompetesIn} \bowtie_{\text{league}} \text{LeagueStadium}) \bowtie_{\text{team, stadium}} (\text{BasedIn} \bowtie_{\text{city}} \text{IsIn})$$

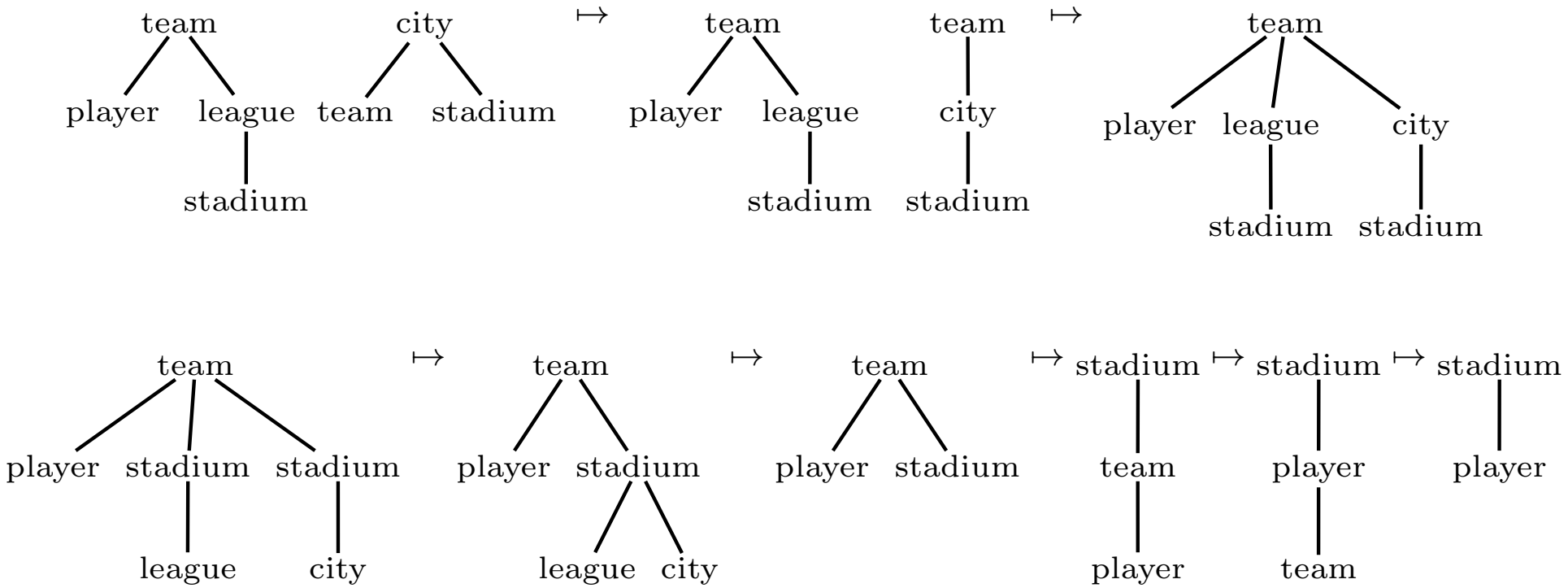


$$\Big(\langle \mathit{FCBarcelona}\rangle \times (\langle \mathit{Messi}\rangle \cup \langle \mathit{Villa}\rangle) \times \times (\langle \mathit{Primera}\rangle \times \langle \mathit{CampNou}\rangle \cup \langle \mathit{Champions}\rangle \times (\langle \mathit{CampNou}\rangle \cup \langle \mathit{Wembley}\rangle)) \cup \langle \mathit{Chelsea}\rangle \times (\langle \mathit{Cech}\rangle \cup \langle \mathit{Torres}\rangle) \times \times (\langle \mathit{Premier}\rangle \times (\langle \mathit{Stamford}\rangle \cup \langle \mathit{Wembley}\rangle)) \cup \langle \mathit{Champions}\rangle \times (\langle \mathit{CampNou}\rangle \cup \langle \mathit{Wembley}\rangle)) \cup \langle \mathit{Arsenal}\rangle \times \langle \mathit{van\ Persie}\rangle \times \times (\langle \mathit{Premier}\rangle \times (\langle \mathit{Stamford}\rangle \cup \langle \mathit{Wembley}\rangle))\Big)$$

$$\times \Big(\langle \mathit{Barcelona}\rangle \times \langle \mathit{FCBarcelona}\rangle \times \langle \mathit{CampNou}\rangle \cup \langle \mathit{London}\rangle \times (\langle \mathit{Chelsea}\rangle \cup \langle \mathit{Arsenal}\rangle) \times (\langle \mathit{Wembley}\rangle \cup \langle \mathit{Stamford}\rangle)\Big)$$

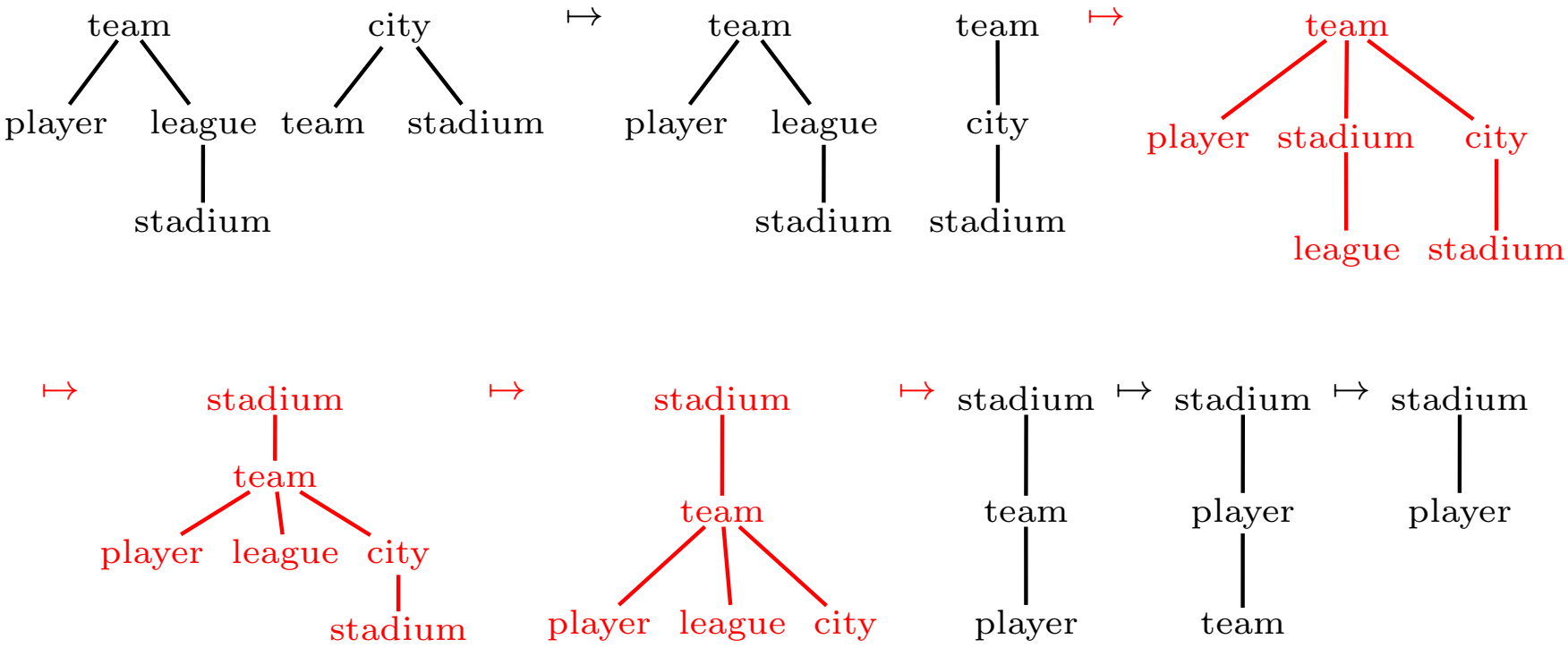
Now perform selections $\mathit{team} = \mathit{team}$ and $\mathit{stadium} = \mathit{stadium}$.

Query Evaluation: Example F-plan



$$\langle CampNou \rangle \times (\langle Messi \rangle \cup \langle Villa \rangle) \cup$$
$$\langle Wembley \rangle \times (\langle Cech \rangle \cup \langle Torres \rangle \cup \langle van\ Persie \rangle) \cup$$
$$\langle Stamford \rangle \times (\langle Cech \rangle \cup \langle Torres \rangle \cup \langle van\ Persie \rangle)$$

Query Evaluation: Different Example F-plan



$$\langle CampNou \rangle \times (\langle Messi \rangle \cup \langle Villa \rangle) \cup$$
$$\langle Wembley \rangle \times (\langle Cech \rangle \cup \langle Torres \rangle \cup \langle van\ Persie \rangle) \cup$$
$$\langle Stamford \rangle \times (\langle Cech \rangle \cup \langle Torres \rangle \cup \langle van\ Persie \rangle)$$

Query Optimisation: Challenges

How to choose a good f-plan for a given query?

New challenges:

- New degrees of freedom:
  Restructuring between joins.
- New (conflicting) objectives:
  Fast execution vs. compact representation of the result?
- How to estimate factorisation size in advance?

Query Optimisation: Approaches

Exhaustive search:

- Explore the entire space of allowed operators, find cheapest path.

Greedy search:

- Always choose the cheapest query operator.
- Always choose the cheapest restructuring to enable query operators.

Metrics for factorisation size and evaluation time:

- Asymptotic size measure $s(\mathcal{T})$.
- Estimates from statistical catalogue information about the data.

FDB

We implement all these ideas in

FDB = Prototype DB engine using factorisations at storage level.
(In-memory, written in C++)

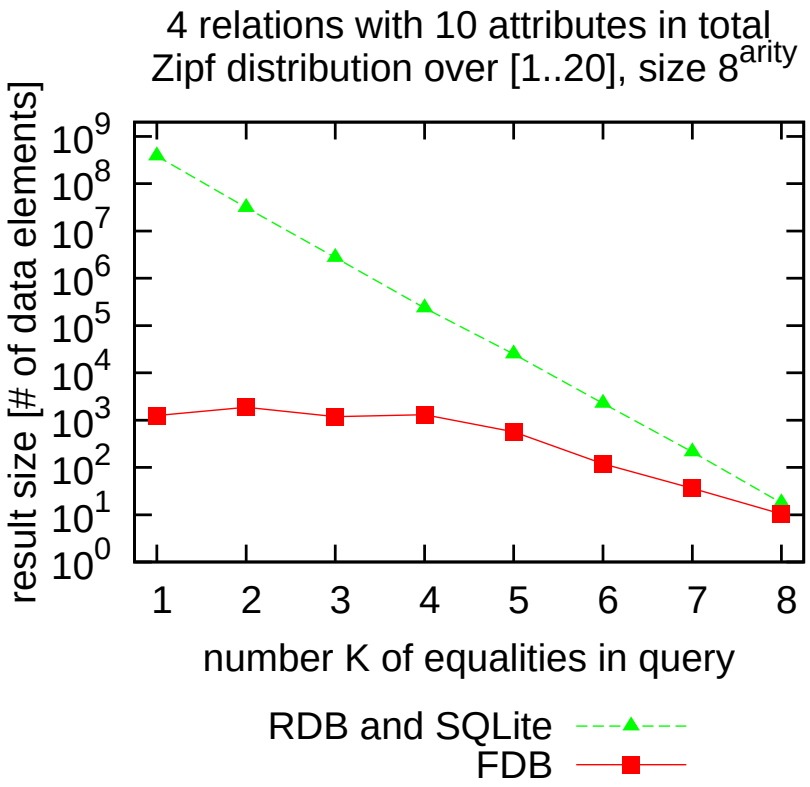
Experimentally compared with

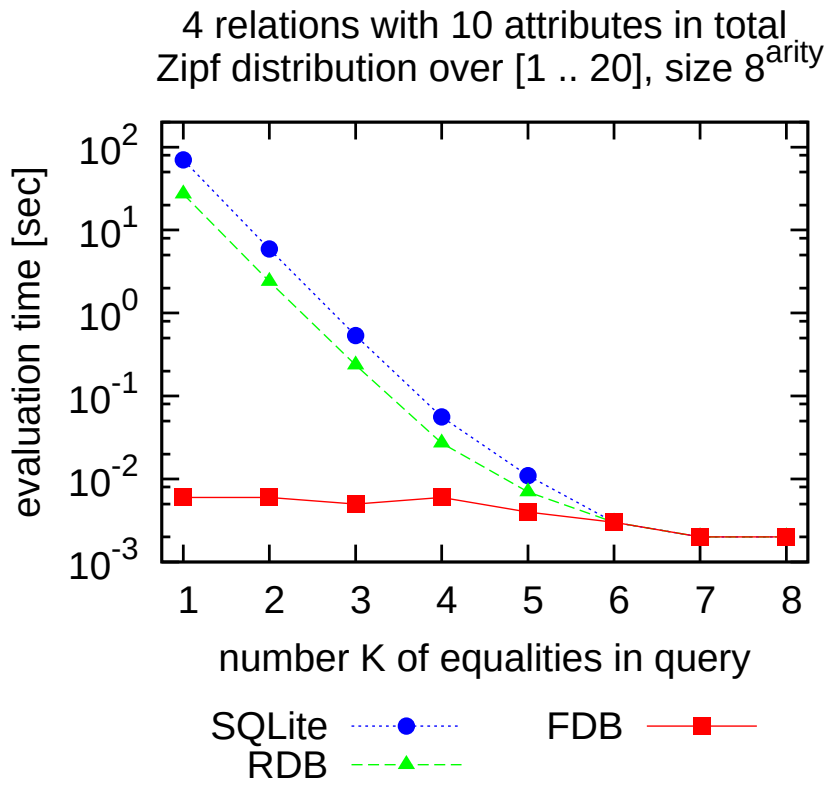
RDB = Lightweight relational engine.
(In-memory, written in C++)
SQLite
PostgreSQL

Experimental Evaluation

FDB vs. RDB vs. SQLite

Queries with K equalities. Input is flat relations.

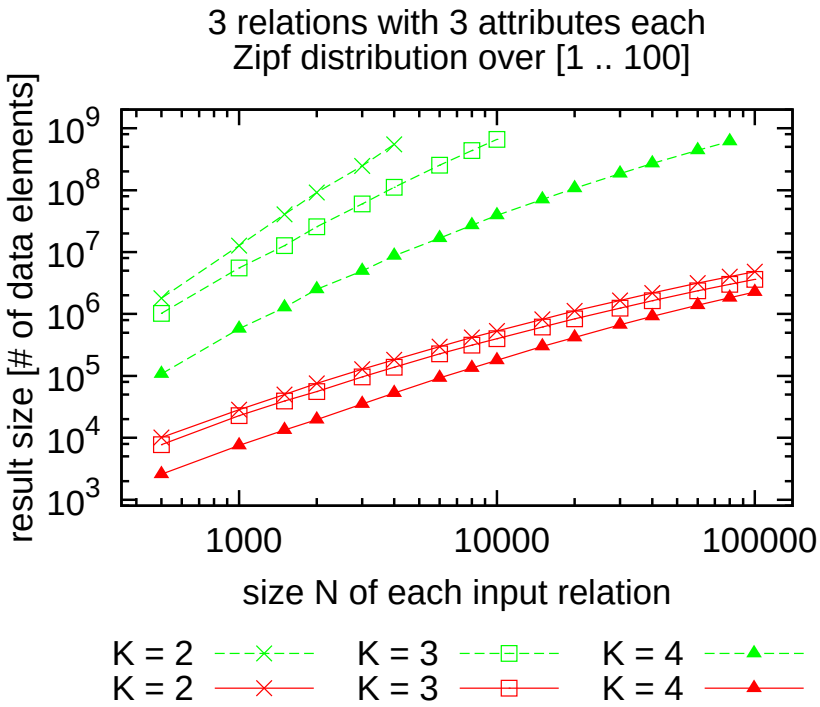


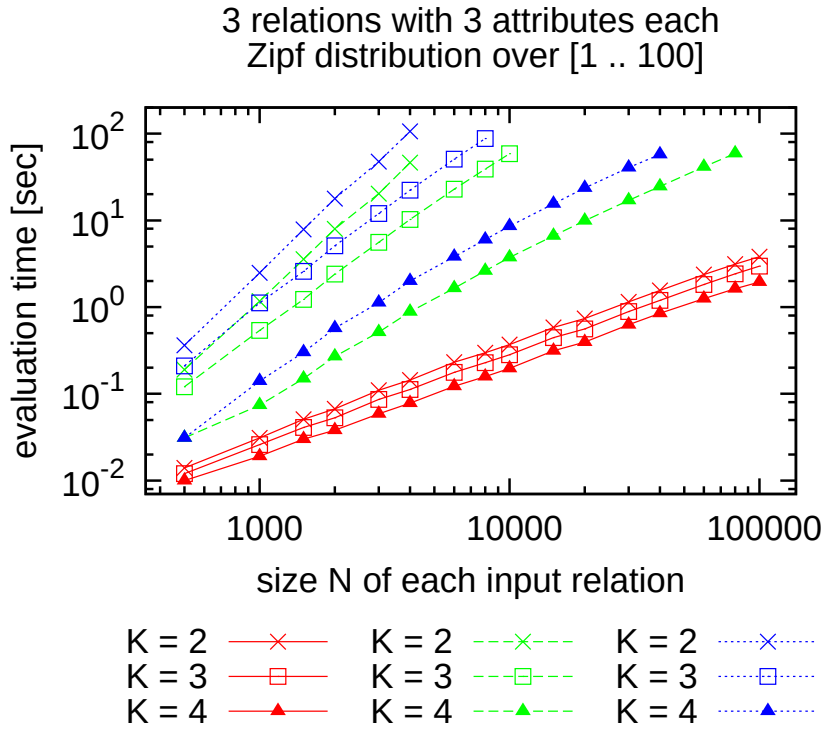


Experimental Evaluation

FDB vs. RDB vs. SQLite

Scaling behaviour with increasing data size. Input is flat relations.





Recent Development: Aggregation, Ordering

"Find the total price of items sold."

```
SELECT SUM(price) FROM Sales;
```

"Find the average price of items sold to each customer."

```
SELECT customer,AVG(price) FROM Sales
    GROUP BY customer;
```

"Find the customers with top 10 sales."

```
SELECT customer,SUM(price) AS spending FROM Sales
    GROUP BY customer
    ORDER BY spending
    LIMIT 10;
```

Aggregation on Factorisations

Aggregation can also be done in linear time, using recursion:

The number of tuples in a factorisation F, $\mathrm{COUNT}(F)$, is defined as:

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

Sum of all values of attribute A in a factorisation F, $\mathrm{SUM}_A(F)$, is defined as:

- $\mathrm{SUM}_A(\langle a \rangle) = a$,
  if the value $\langle a \rangle$ is of attribute A.
- $\mathrm{SUM}_A(F_1 \cup \cdots \cup F_k) = \mathrm{SUM}_A(F_1) + \ldots + \mathrm{SUM}_A(F_k)$.
- $\mathrm{SUM}_A(F_1 \times \cdots \times F_k) = \mathrm{SUM}_A(F_1) \cdot \mathrm{COUNT}(F_2) \cdot \ldots \cdot \mathrm{COUNT}(F_k)$,
  where wlog values of attribute A are in expression F_1.

Aggregation: Example

Orders

customer	date	pizza
Mario	Monday	Capricciosa
Mario	Tuesday	Margherita
Pietro	Friday	Hawaii
Lucia	Friday	Hawaii
Mario	Friday	Capricciosa

Pizzas

pizza	item
Margherita	base
Capricciosa	base
Capricciosa	ham
Capricciosa	mushrooms
Hawaii	base
Hawaii	ham
Hawaii	pineapple

Items

item	price
base	6
ham	1
mushrooms	1
pineapple	2

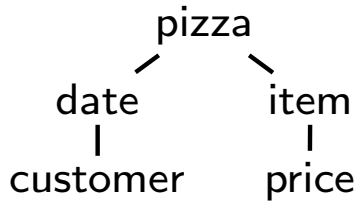
Itemized sales: $\mathrm{Sales} = \mathrm{Orders} \bowtie \mathrm{Pizzas} \bowtie \mathrm{Items}$.

Revenue per customer: $\varpi_{customer, \mathrm{sum}_{price}}(\mathrm{Sales})$.

```
SELECT customer, SUM(price) FROM Sales GROUP BY customer;
```

Aggregation: Example

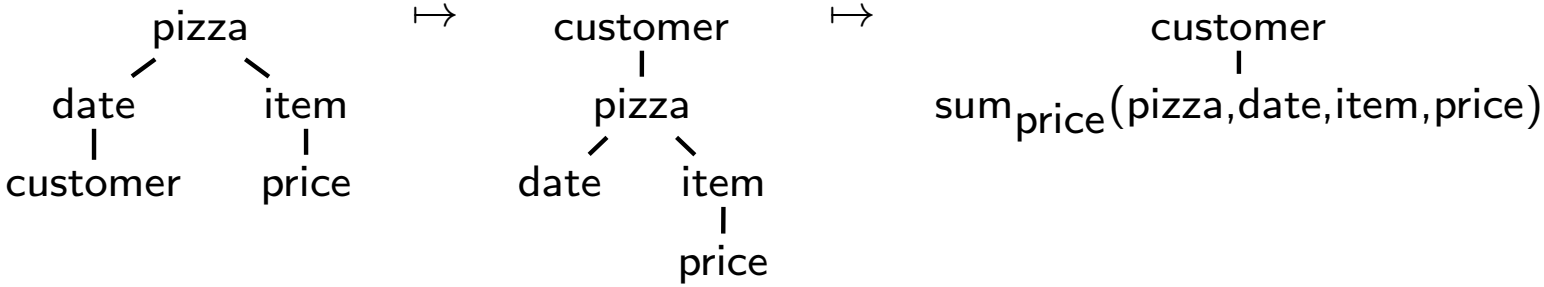
$Sales = \text{Orders} \bowtie \text{Pizzas} \bowtie \text{Items}$



$$
\begin{aligned}
&\langle \mathit{Capricciosa} \rangle \times (\langle \mathit{Monday} \rangle \times \langle \mathit{Mario} \rangle \cup \langle \mathit{Friday} \rangle \times \langle \mathit{Mario} \rangle) \\
&\qquad \times (\langle \mathit{base} \rangle \times \langle 6 \rangle \cup \langle \mathit{ham} \rangle \times \langle 1 \rangle \cup \langle \mathit{mushrooms} \rangle \times \langle 1 \rangle) \\
&\cup \langle \mathit{Hawaii} \rangle \times \langle \mathit{Friday} \rangle \times (\langle \mathit{Lucia} \rangle \cup \langle \mathit{Pietro} \rangle) \\
&\qquad \times (\langle \mathit{base} \rangle \times \langle 6 \rangle \cup \langle \mathit{ham} \rangle \times \langle 1 \rangle \cup \langle \mathit{pineapple} \rangle \times \langle 2 \rangle) \\
&\cup \langle \mathit{Margherita} \rangle \times \langle \mathit{Tuesday} \rangle \times \langle \mathit{Mario} \rangle \times \langle \mathit{base} \rangle \times \langle 6 \rangle
\end{aligned}
$$

Aggregation: Example

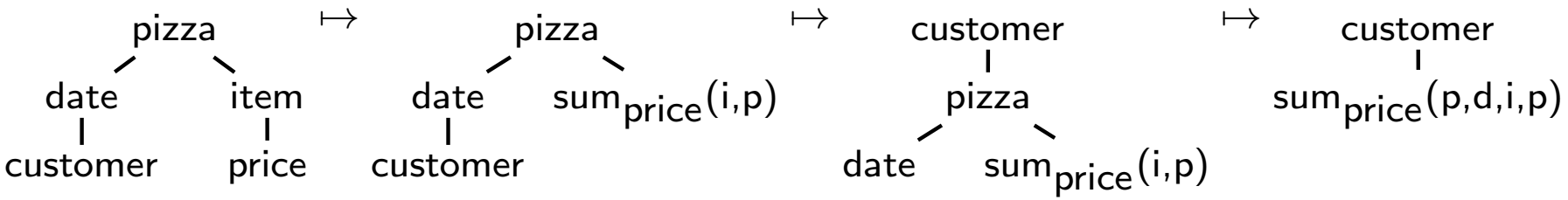
$\varpi_{customer,\mathrm{sum}_{price}}(\mathrm{Sales})$



$$\begin{aligned}
&\langle Capricciosa\rangle \times (\langle Monday\rangle \times \langle Mario\rangle \cup \langle Friday\rangle \times \langle Mario\rangle)\\
&\qquad\qquad \times (\langle base\rangle \times \langle 6\rangle \cup \langle ham\rangle \times \langle 1\rangle \cup \langle mushrooms\rangle \times \langle 1\rangle)\\
&\cup \langle Hawaii\rangle \times \langle Friday\rangle \times (\langle Lucia\rangle \cup \langle Pietro\rangle)\\
&\qquad\qquad \times (\langle base\rangle \times \langle 6\rangle \cup \langle ham\rangle \times \langle 1\rangle \cup \langle pineapple\rangle \times \langle 2\rangle)\\
&\cup \langle Margherita\rangle \times \langle Tuesday\rangle \times \langle Mario\rangle \times \langle base\rangle \times \langle 6\rangle
\end{aligned}$$

Aggregation: Example

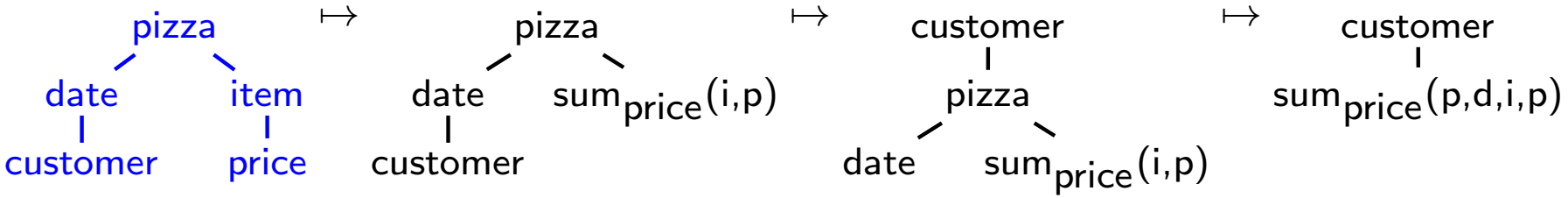
$\varpi_{customer,\mathrm{sum}_{price}}(\mathrm{Sales})$



$$\begin{aligned}
&\langle Capricciosa\rangle \times (\langle Monday\rangle \times \langle Mario\rangle \cup \langle Friday\rangle \times \langle Mario\rangle)\\
&\qquad\qquad \times (\langle base\rangle \times \langle 6\rangle \cup \langle ham\rangle \times \langle 1\rangle \cup \langle mushrooms\rangle \times \langle 1\rangle)\\
&\cup \langle Hawaii\rangle \times \langle Friday\rangle \times (\langle Lucia\rangle \cup \langle Pietro\rangle)\\
&\qquad\qquad \times (\langle base\rangle \times \langle 6\rangle \cup \langle ham\rangle \times \langle 1\rangle \cup \langle pineapple\rangle \times \langle 2\rangle)\\
&\cup \langle Margherita\rangle \times \langle Tuesday\rangle \times \langle Mario\rangle \times \langle base\rangle \times \langle 6\rangle
\end{aligned}$$

Aggregation: Example

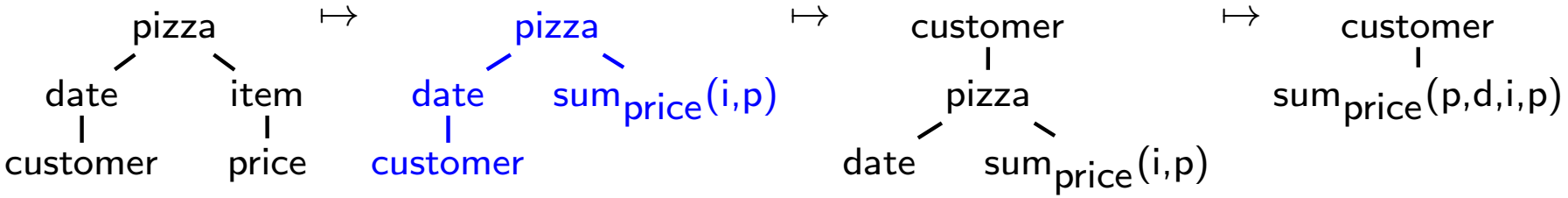
$\varpi_{customer,\mathrm{sum}_{price}}(\mathrm{Sales})$



$$\begin{aligned}
&\langle \mathit{Capricciosa}\rangle \times (\langle \mathit{Monday}\rangle \times \langle \mathit{Mario}\rangle \cup \langle \mathit{Friday}\rangle \times \langle \mathit{Mario}\rangle)\\
&\qquad\qquad \times (\langle \mathit{base}\rangle \times \langle 6\rangle \cup \langle \mathit{ham}\rangle \times \langle 1\rangle \cup \langle \mathit{mushrooms}\rangle \times \langle 1\rangle)\\
&\cup \langle \mathit{Hawaii}\rangle \times \langle \mathit{Friday}\rangle \times (\langle \mathit{Lucia}\rangle \cup \langle \mathit{Pietro}\rangle)\\
&\qquad\qquad \times (\langle \mathit{base}\rangle \times \langle 6\rangle \cup \langle \mathit{ham}\rangle \times \langle 1\rangle \cup \langle \mathit{pineapple}\rangle \times \langle 2\rangle)\\
&\cup \langle \mathit{Margherita}\rangle \times \langle \mathit{Tuesday}\rangle \times \langle \mathit{Mario}\rangle \times \langle \mathit{base}\rangle \times \langle 6\rangle
\end{aligned}$$

Aggregation: Example

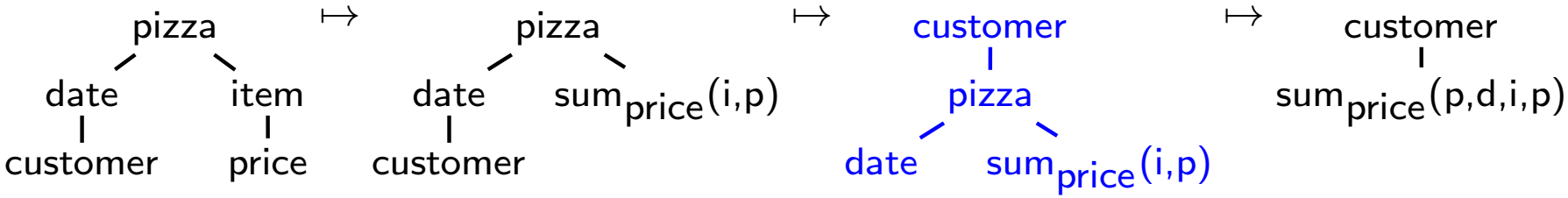
$\varpi_{customer,\mathrm{sum}_{price}}(\mathrm{Sales})$



$$\begin{aligned}
&\langle Capricciosa\rangle \times (\langle Monday\rangle \times \langle Mario\rangle \cup \langle Friday\rangle \times \langle Mario\rangle) \times \langle 8\rangle \\
\cup\, &\langle Hawaii\rangle \times \langle Friday\rangle \times (\langle Lucia\rangle \cup \langle Pietro\rangle) \times \langle 9\rangle \\
\cup\, &\langle Margherita\rangle \times \langle Tuesday\rangle \times \langle Mario\rangle \times \langle 6\rangle
\end{aligned}$$

Aggregation: Example

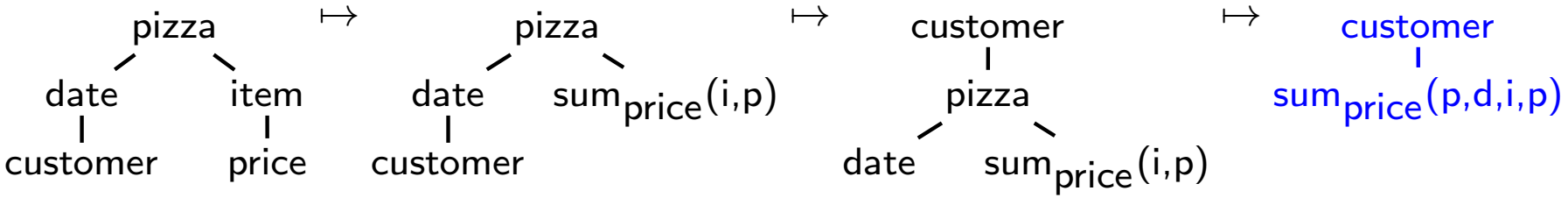
$\varpi_{customer,\mathrm{sum}_{price}}(\mathrm{Sales})$



$$\begin{aligned}
&\langle Lucia\rangle \times \langle Hawaii\rangle \times \langle Friday\rangle \times \langle 9\rangle \\
\cup\, &\langle Mario\rangle \times (\langle Capricciosa\rangle \times (\langle Monday\rangle \cup \langle Friday\rangle) \times \langle 8\rangle \cup \\
&\qquad \langle Margherita\rangle \times \langle Tuesday\rangle \times \langle 6\rangle) \\
\cup\, &\langle Pietro\rangle \times \langle Hawaii\rangle \times \langle Friday\rangle \times \langle 9\rangle
\end{aligned}$$

Aggregation: Example

$\varpi_{customer,\mathrm{sum}_{price}}(\mathrm{Sales})$

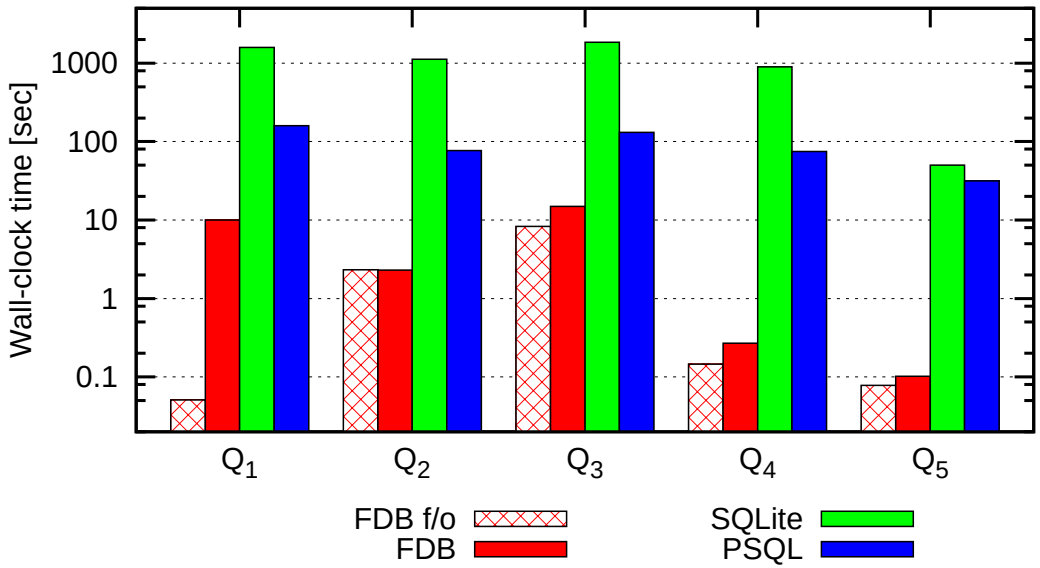


$$\langle Lucia \rangle \times \langle 9 \rangle \cup \langle Mario \rangle \times \langle 22 \rangle \cup \langle Pietro \rangle \times \langle 9 \rangle$$

Experimental Evaluation

FDB vs. SQlite vs. PostgreSQL

Aggregation queries on $Sales$ = Orders $\bowtie$ Items $\bowtie$ Packages.

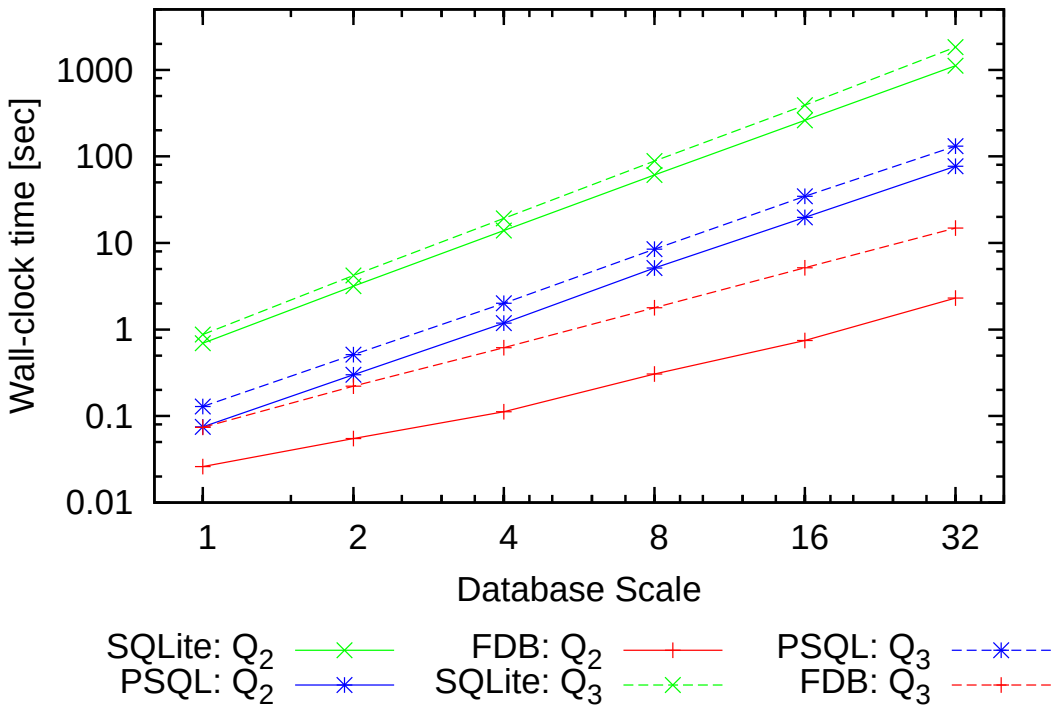


$$Q_1 = \varpi_{\text{package, date, customer; sum(price)}}(Sales)$$
$$Q_2 = \varpi_{\text{customer; revenue} \leftarrow \text{sum(price)}}(Sales)$$
$$Q_3 = \varpi_{\text{date, package; sum(price)}}(Sales)$$
$$Q_4 = \varpi_{\text{package; sum(price)}}(Sales)$$
$$Q_5 = \varpi_{\text{sum(price)}}(Sales)$$

Experimental Evaluation

FDB vs. SQlite vs. PostgreSQL

Aggregation queries on *Sales* = Orders ⋈ Items ⋈ Packages.
Performance with increasing data size.

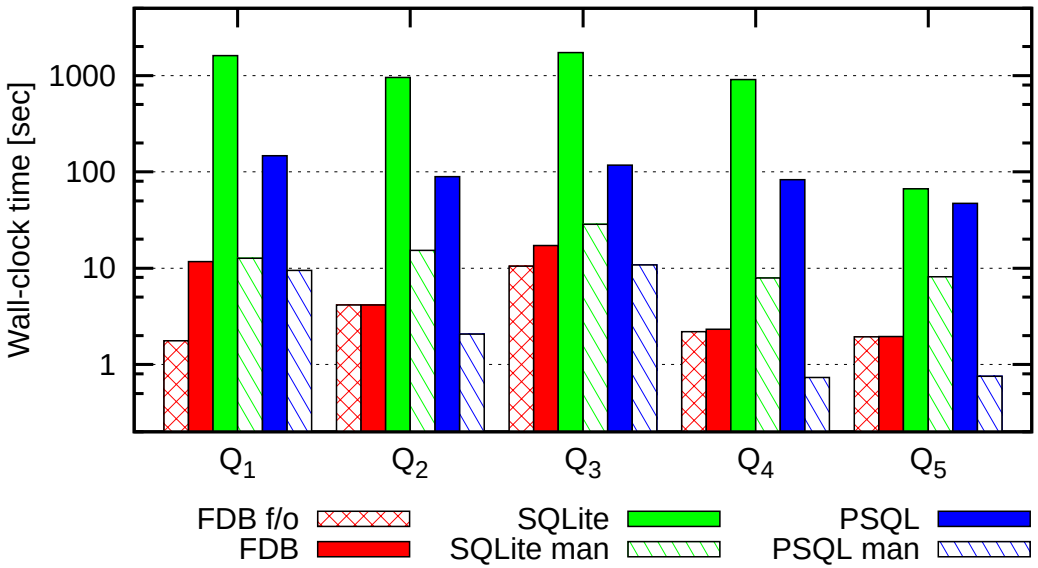


$Q_2 = \varpi_{\text{customer; revenue} \leftarrow \text{sum(price)}}(Sales)$
$Q_3 = \varpi_{\text{date, package; sum(price)}}(Sales)$

Experimental Evaluation

FDB vs. SQlite vs. PostgreSQL

Aggregation queries on Orders, Items and Packages.



$Q_1 = \varpi_{\text{package, date, customer; sum(price)}}(Sales)$
$Q_2 = \varpi_{\text{customer; revenue} \leftarrow \text{sum(price)}}(Sales)$
$Q_3 = \varpi_{\text{date, package; sum(price)}}(Sales)$
$Q_4 = \varpi_{\text{package; sum(price)}}(Sales)$
$Q_5 = \varpi_{\text{sum(price)}}(Sales)$

Experimental Evaluation

Factorisations **can boost query performance** for many-to-many data.

FDB outperforms relational engines SQlite and PostgreSQL:

- if input is factorised.
- if output can be factorised.
- for flat to flat aggregation.
  (Unless they are hand-optimised to push aggregates past joins.)
- for flat to flat queries with `ORDER BY` and `LIMIT`.

Factorised Relations

$$\Big(\langle \mathit{Tesco}\rangle \times (\langle \mathit{food}\rangle \cup \langle \mathit{clothes}\rangle) \cup \langle \mathit{Lidl}\rangle \times \langle \mathit{food}\rangle\Big) \times (\langle \mathit{CzechRep}\rangle \cup \langle \mathit{Slovakia}\rangle) \cup$$
$$(\langle \mathit{Tesco}\rangle \cup \langle \mathit{Lidl}\rangle \cup \langle \mathit{Aldi}\rangle) \times \langle \mathit{food}\rangle \times (\langle \mathit{Poland}\rangle \cup \langle \mathit{UK}\rangle) \cup$$
$$(\langle \mathit{Tesco}\rangle \cup \langle \mathit{Aldi}\rangle) \times \langle \mathit{books}\rangle \times (\langle \mathit{UK}\rangle \cup \langle \mathit{Spain}\rangle)$$

Representations of relations that

- compactly encode combinations of groups of values.
- allow for enumerating tuples with constant delay.
- allow query evaluation *without expanding* to flat relations
  - Using factorisations can boost query evaluation (processing a smaller representation).

Thank you!