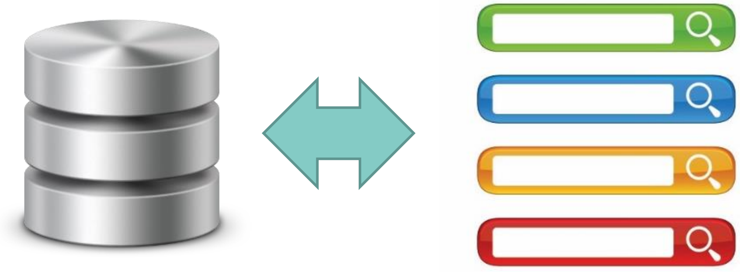
DrillBeyond: Processing Multi-Result Open World SQL Queries

Julian Eberius*, Maik Thiele, Katrin Braunschweig and Wolfgang Lehner

Technische Universität Dresden, Germany

Motivation

Dresden Database Systems Group



DOMAINS OF…

- …information retrieval and
- database systems…

…HAVE BEEN TRADITIONALLY KEPT SEPARATE

DIFFERENCES IN…

▪ type of data that is managed	(structured	versus	unstructured)
▪ query language used	(fully specified query	versus	keyword query)
▪ nature of the query result	(exact single answer	versus	ranked list of possible answers)
▪ usage scenarios	(analytic scenarios	versus	information gathering)

WE EXPLORE A NEW WAY…

- …of merging the two paradigms for use in ad-hoc and self-service analytics

DrillBeyond

Dresden Database Systems Group

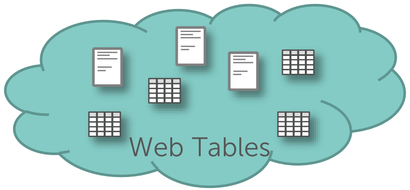
DrillBeyond = Novel hybrid DBMS/IR system

- Blurring the lines between type of data managed, query language used, and nature of the query result

```
select n_name, gdp , avg(o_totalprice)
from nation, customer, orders
where
    n_nationkey=c_nationkey
    andc_custkey=o_custkey
    and gdp > 10.0
group by n_name, gdp
order by gdp desc
```

Mix of...

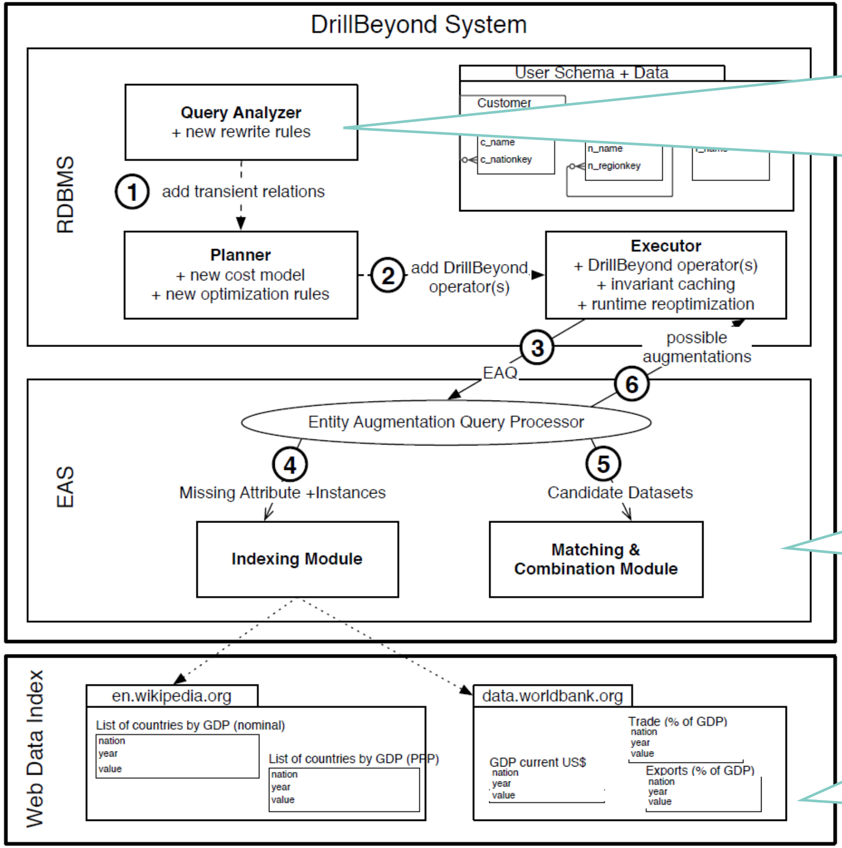
- ...relational queries on a database and...
- ...top-k keyword-based searches (= Enity Augmentation Queries)



DrillBeyond – System Architecture

Dresden Database Systems Group

talk yesterday



Query Analyzer

- Maps SQL query tokens to the database catalog
- For unrecognized tokens (e.g. "gdp") we introduce transient metadata
- Query is rewritten to include an additional join with a transient relation

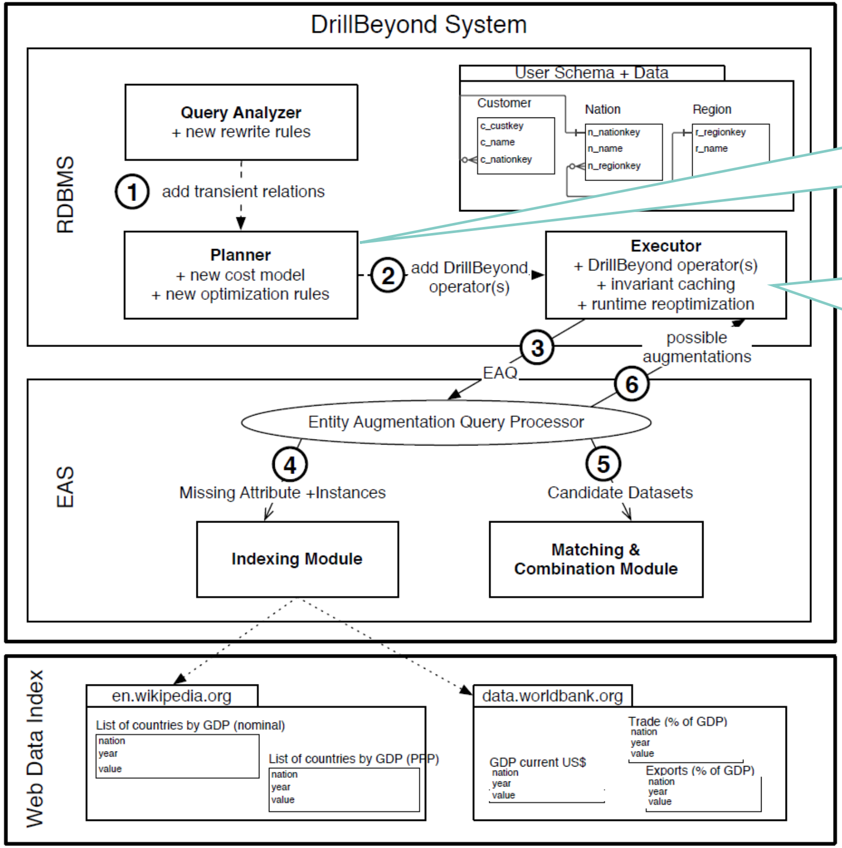
Entity Augmentation System

- Implements the entity augmentation processing

Web Data Index

- Index of Web tables
- Generic system exposing an interface for keyword-based document search

DrillBeyond – System Architecture (2)



Query Planner

- Minimize the overhead of creating multiple result variants

Executor

- Implements the repeated execution of operator trees using the DrillBeyond operator ω

System Integration Challenges

Dresden Database Systems Group

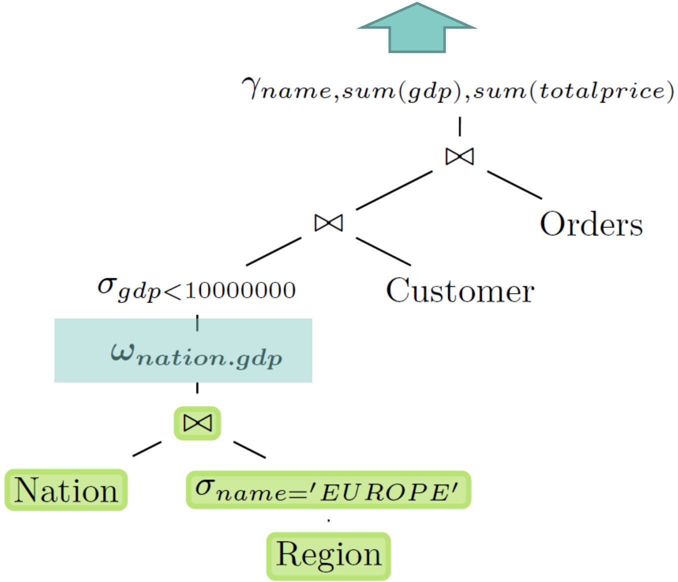
Multi-solution Query Processing

- Top-k query result versions for a single user query
- Naïve way: process the query k times → query runtime increased by factor k
- Goals:
  - **Minimizes duplicate work** between query executions
  - **Maximize invariant parts** of multiple executions

Query Planning

- Web tables are not fully known at plan-time
  → no selectivity information or data types at plan-time
- Goal: **plan queries** with relations only known **at run-time**

n_name	gdp	total
US	313.2	43.8
Germany	81.4	29.1
France	65.3	27.5



DrillBeyond Operator ω

Init()

- Inititializes state

$$
\begin{array}{l}
\textbf{function } \textsc{Init} \\
\quad state \leftarrow \text{`}collecting\text{'} \\
\quad tuplestore \leftarrow \emptyset \\
\quad augMap \leftarrow HashMap() \\
\quad n \leftarrow 0
\end{array}
$$

Next()

- Produces augmented tupels in three phase
  - Collect()
  - Augment()
  - Project()

$$
\begin{array}{l}
\textbf{function } \textsc{Next} \\
\quad \textbf{if } state = \text{`}collecting\text{'} \textbf{ then} \\
\qquad \textsc{Collect}() \\
\qquad \textsc{Augment}() \\
\qquad state \leftarrow \text{`}projecting\text{'} \\
\quad \textbf{return } \textsc{Project}()
\end{array}
$$

DrillBeyond Operator ω (2)

COLLECT()

- Pulls and stores all tuples
- Blocking

$$
\begin{array}{l}
\textbf{function } \textsc{Collect} \\
\quad \textbf{while } \mathit{true} \textbf{ do} \\
\quad\quad t \leftarrow \textsc{Next}(\mathit{childPlan}) \\
\quad\quad \textbf{if } t = \mathit{NULL} \textbf{ then} \\
\quad\quad\quad \textbf{break} \\
\quad\quad \mathit{tuplestore} \leftarrow t \\
\quad\quad \mathit{augKey} \leftarrow \textsc{textAttrs}(t) \\
\quad\quad \textbf{if } \mathit{augKey} \notin \mathit{augMap} \textbf{ then} \\
\quad\quad\quad \mathit{augMap}[\mathit{augKey}] \leftarrow \emptyset
\end{array}
$$

AUGMENT()

- Pass all entries from the EAS into a hashtable

$$
\begin{array}{l}
\textbf{function } \textsc{Augment} \\
\quad \mathit{augReq} \leftarrow (\forall k \in \mathit{augMap} \mid \mathit{augMap}[k] = \emptyset) \\
\quad \textbf{for all } \mathit{augKey}, [\mathit{augValues}...] \in \textsc{Send}(\mathit{augReq}) \textbf{ do} \\
\quad\quad \mathit{augMap}[\mathit{augKey}] \leftarrow [\mathit{augValues}...]
\end{array}
$$

DrillBeyond Operator ω (3)

PROJECT()

- Produces output tuples by replaying the stored tuples and filling the augmentation attribute

$$\begin{array}{l}\textbf{function } \textsc{Project} \\ \quad t \leftarrow \textsc{Next}(tuplestore) \\ \quad \textbf{if } t = NULL \textbf{ then return } NULL \\ \quad augKey \leftarrow \textsc{textAttrs}(t) \\ \quad t[augAttr] = augMap[augKey][n] \textbf{ return } t\end{array}$$

RESCAN()

- Called when subtrees have to be re-executed

$$\begin{array}{l}\textbf{function } \textsc{Rescan} \\ \quad state \leftarrow \text{'}collecting\text{'} \\ \quad tuplestore \leftarrow \emptyset\end{array}$$

NEXTVARIANT()

- Produces the multi-variant query results

$$\begin{array}{l}\textbf{function } \textsc{NextVariant} \\ \quad \textsc{ReScan}(tuplestore) \\ \quad n \leftarrow n + 1\end{array}$$

Why Blocking?

Dresden Database Systems Group

country	gdp
Russia	?
UK	?
USA	?

Example

- Send each tuple to the augmentation system at is own (tuple-at-a-time)
- Augmentation system may choose ds_1, ds_2 and ds_4

More consistent: DS_1 and DS_3

For quality reasons the DrillBeyond operator…

- …needs to be a **blocking** operator
- Realized in the "collecting" state in function Next()
- Consumes tuples from underlying operators until these are exhausted
- Hands them over to the augmentation system

ds_1 American Countries

country	gdp m USD
USA	X
Canada	Y

ds_3 World GDPs

country	gdp Mil. $
UK	X
Russia	Z

ds_2 European Economy

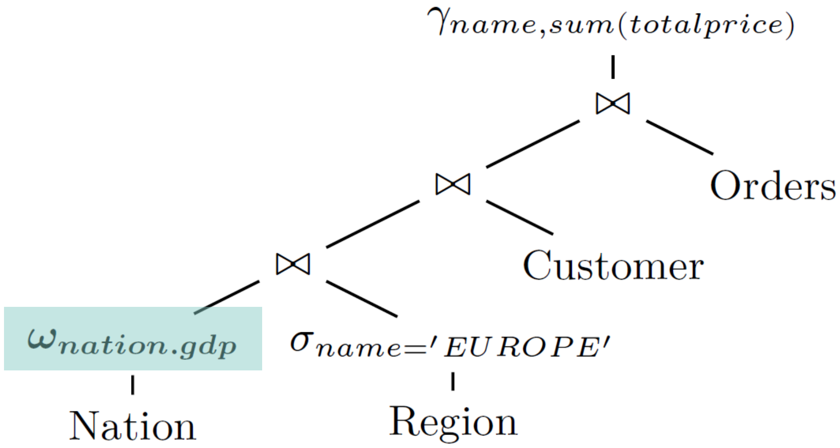
country	gdp EUR
Russia	X
France	Y

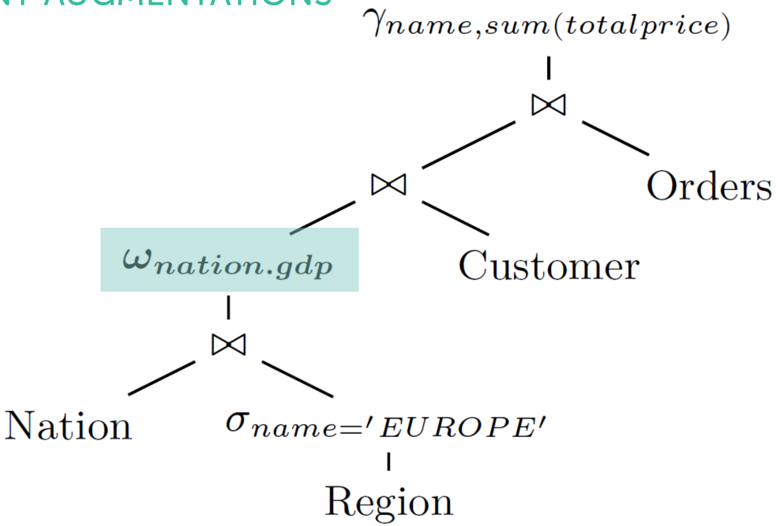
ds_4 European Economy

country	gdp EUR
UK	X

Lower Placement Bound

SAME QUERY, TWO DIFFERENT QUERY PLANS → TWO DIFFERENT AUGMENTATIONS





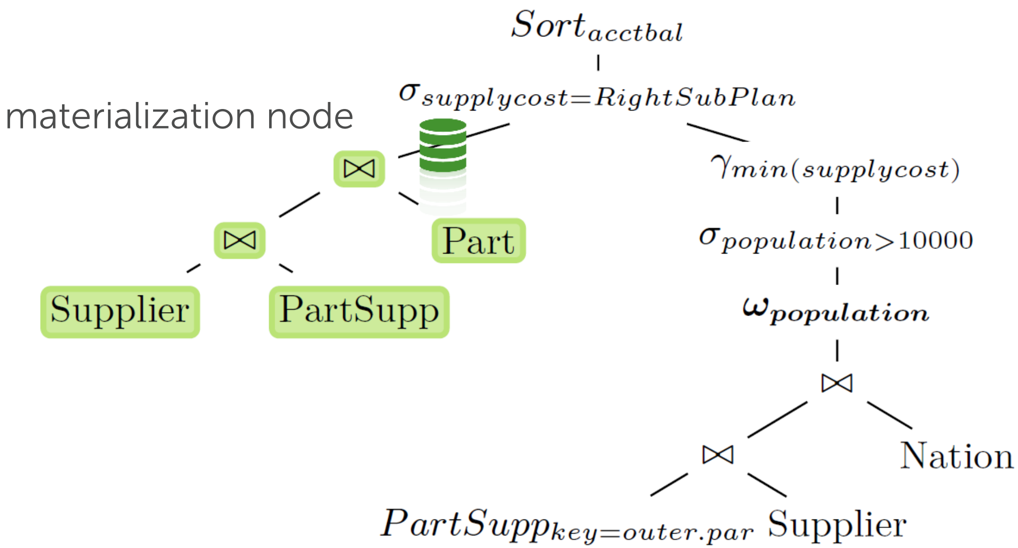
LOWER PLACEMENT BOUND

- ω can only be placed when all filters on R, e.g., joins and predicates, have been applied
- In other words: apply ω to the minimum number of entities in R

Invariant Caching

EXAMPLE

- Modeled after TPCH query 2

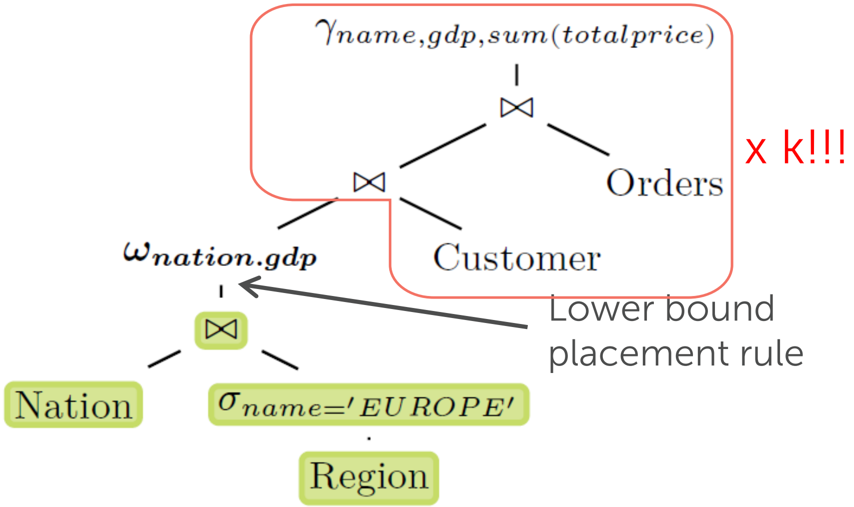


LIMITATION: DEPENDING ON THE COST-BASED PLACEMENT OF ω, MOST OF THE QUERY MAY BE VARYING

Maximize Invariant Sub-Trees

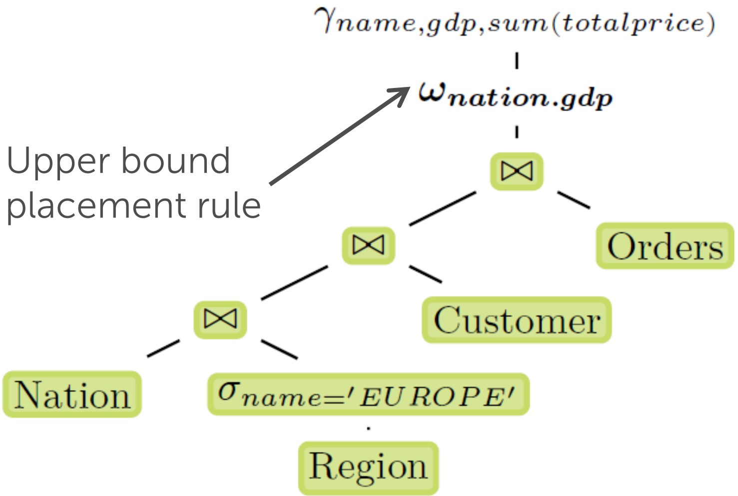
OPTIMAL PLAN…

- …with respect to a single query execution



OPTIMAL PLAN…

- …with respect to multiple query executions
- Upper bound placement rule: Place ω not earlier than the augmented values need to be accessed



Maximize Invariant Sub-Trees (2)

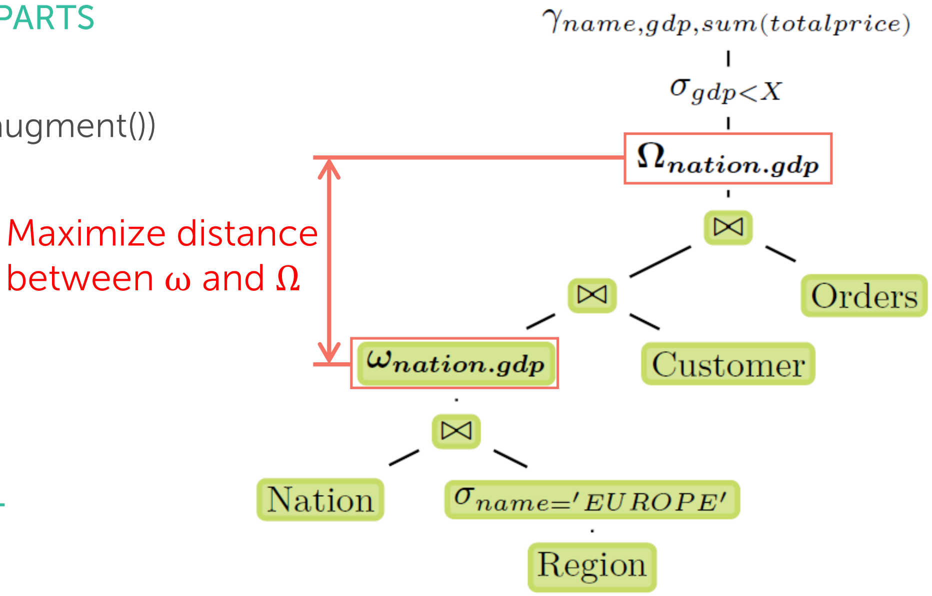
Dresden Database Systems Group

Observation

- We can separate the input part of ω from the actual projection of augmented values

Split the DrillBeyond operator into two parts

- ω
  - Performing the augmentation (collect() and augment())
  - Projects placeholders
  - Placed at the lower placement bound
- Ω
  - Performing the projection of values
  - Dereferences to the correct value array
  - Placed at the upper placement bound



→ MINIMIZING AUGMENTATION OPERATOR COST

→ MAXIMIZING SIZE OF INVARIANT QUERY PARTS

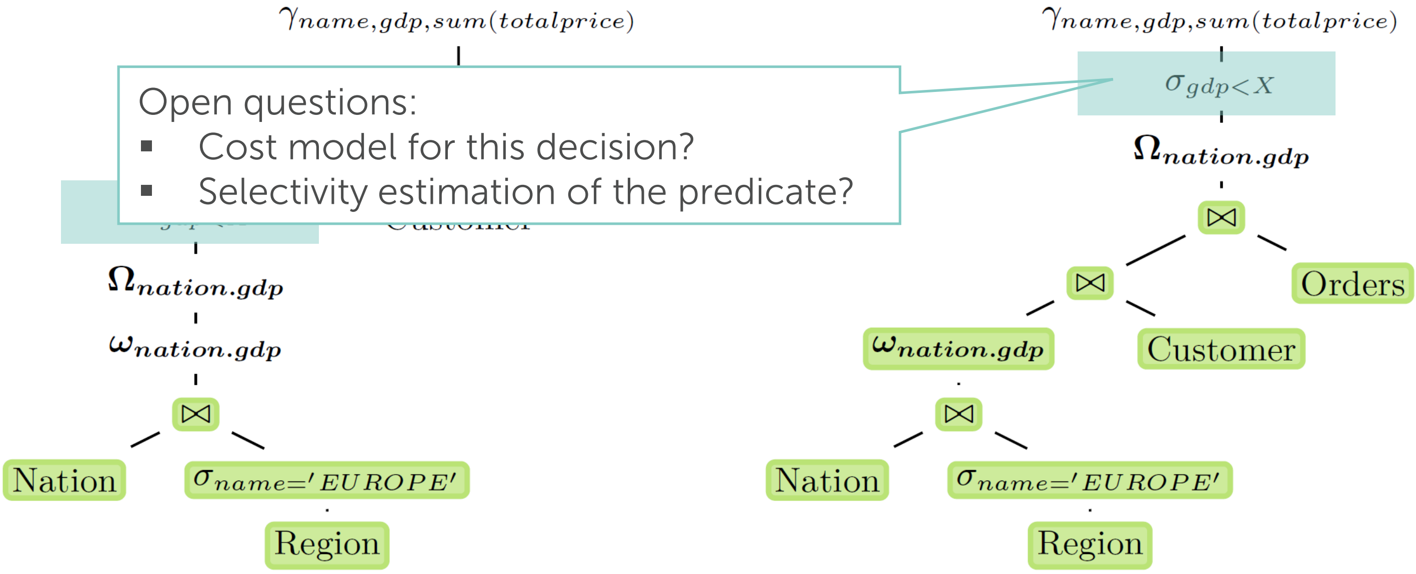
Dynamic Selection Pull-Up

Selection Push-Down

- Smaller intermediate join result…
- …but more varying nodes in the plan

Selection Pull-Up

- Place Ω as describe before
- Larger invariant subplan…
- …but increased join results



Dynamic Selection Pull-Up (2)

Dresden Database Systems Group

Cost Model

- Binary decision → decide for the minimum cost of the subtree above ω

$$C_k(\omega,\mathrm{T}) = \begin{cases} C_1(\omega,\Omega) + \sum_{i=1}^{k} s_i \cdot C_1(\omega,\mathrm{T}) & (Pull-up) \\ \sum_{i=1}^{k} s_i \cdot C_1(\omega,\mathrm{T}) & (no\ Pull-up) \end{cases}$$

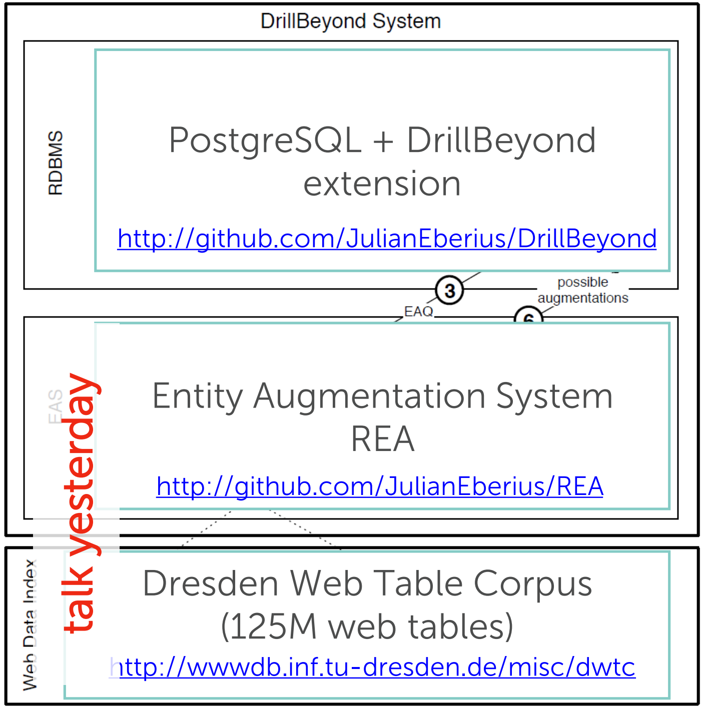
- $C(x,y)$: cost of the subplan from operator x to y
- s_i
  - Selectivity of the i[th] data source for the attribute in question
  - Not known at plan-time → determined in the augment() phase

Evaluation

Experimental Setup

Dresden Database Systems Group

Implementation



Test database

- Variation of TPC-H replacing generic identifiers with real-world entities, e.g. real company names for the Supplier relation

Test queries

- Subset of TPC-H queries in which dimension tables are used
- Added arbitrary where-clauses to one of the dimensions: "relation.X > Y"

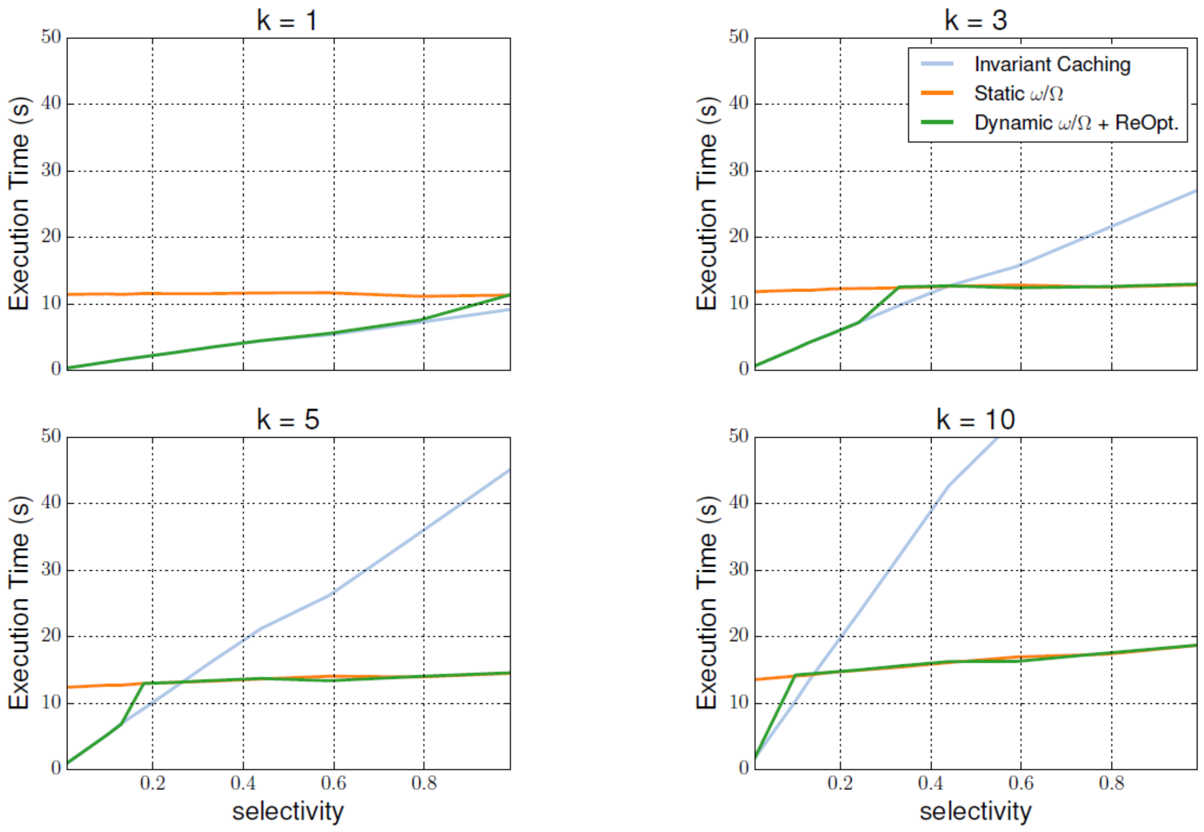
Parameters

- Number of augmentations k ∈ (1, 3, 5, 10)
- Selectivity s ranging between 0.01 and 0.99 in ten steps

Overall Performance

Scenario	Norm. Execution Time	Standard Deviation
No Optimizations	2.27	1.82
Invariant Caching	1.99	1.59
Static ω/Ω	1.79	2.35
Dynamic ω/Ω	1.21	0.67
Dynamic ω/Ω + Re-Opt.	1.13	0.28

Performance of query 9, by selectivity



Summary

Dresden Database Systems Group

DrillBeyond a RDBMS / IR hybrid system

- Integration of top-k entity augmentation system into a RDBMS
- Multiple alternative query result solving the uncertainty and ambiguity of the automated integration

New operators: DrillBeyond ω/Ω

- Implemented in traditional interface functions: Init(), Next() and ReScan()
- Own cost model

New optimization strategies

- Invariant Caching
- Maximization of invariant sub-trees
- Selection push-down versus selection pull-up (runtime reoptimization)