TECHNISCHE UNIVERSITÄT DRESDEN

Dresden Database Systems Group

DrillBeyond: Processing Multi-Result Open World SQL Queries

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

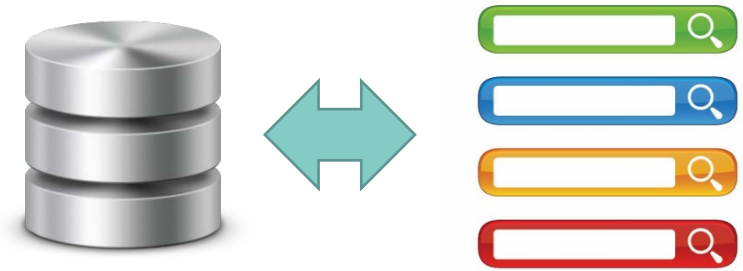
Technische Universität Dresden, Germany

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

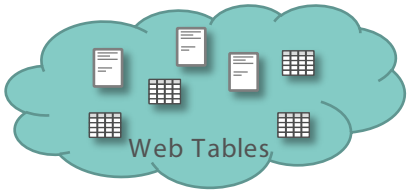
DrillBeyond = Novel hybrid DBMS/IR system

- 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
```

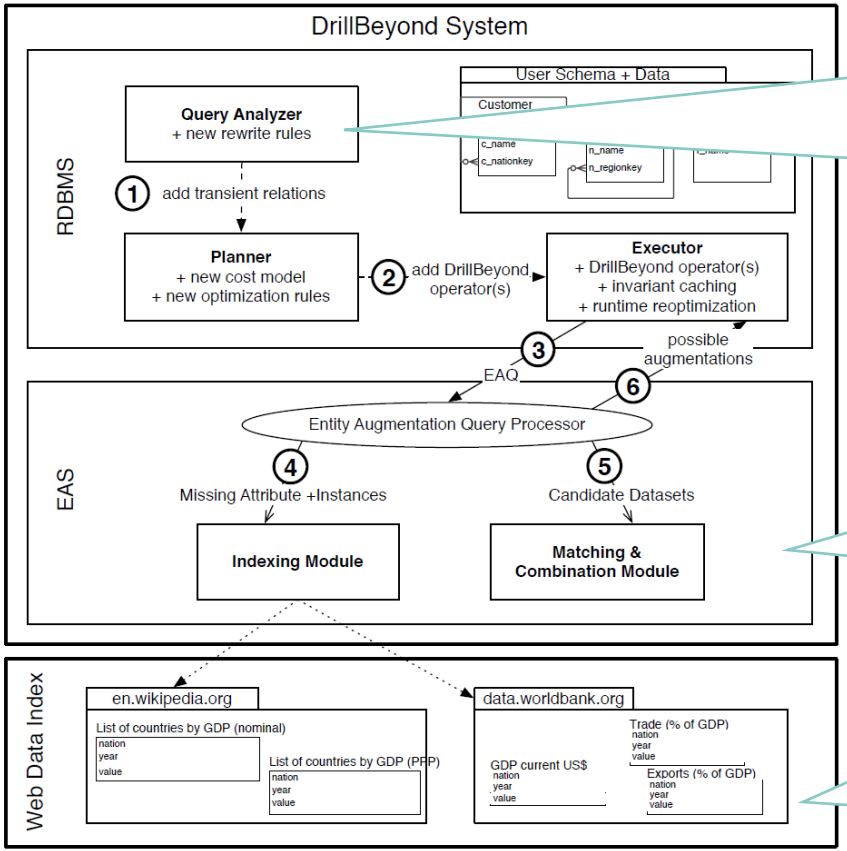
Mix of...

- ..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

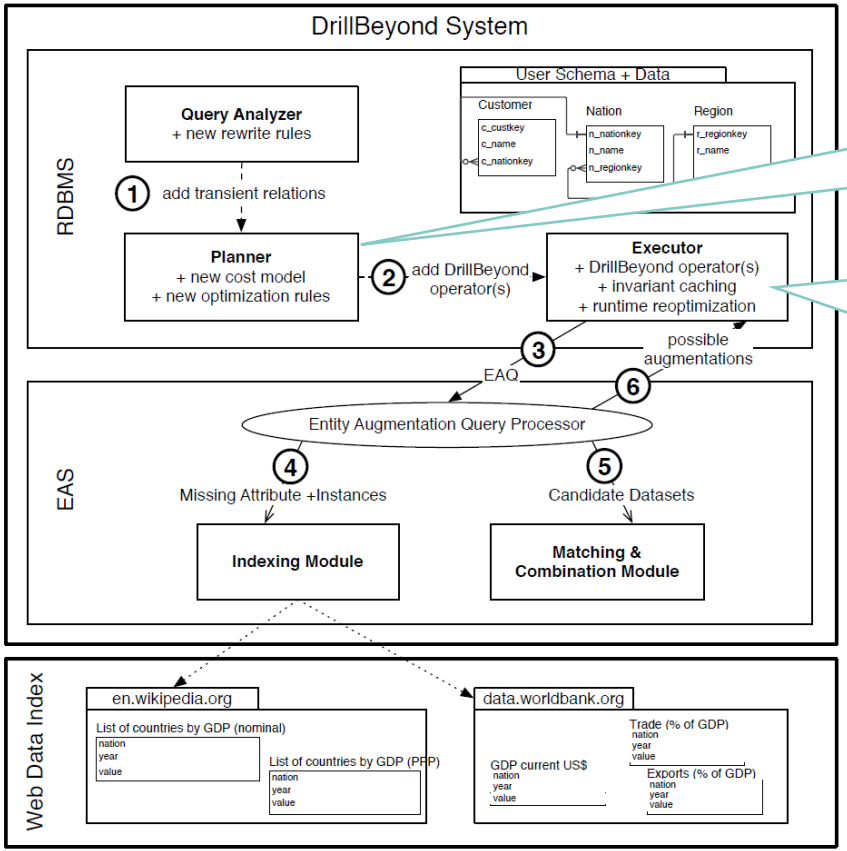
Entity Augmentation System

- 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

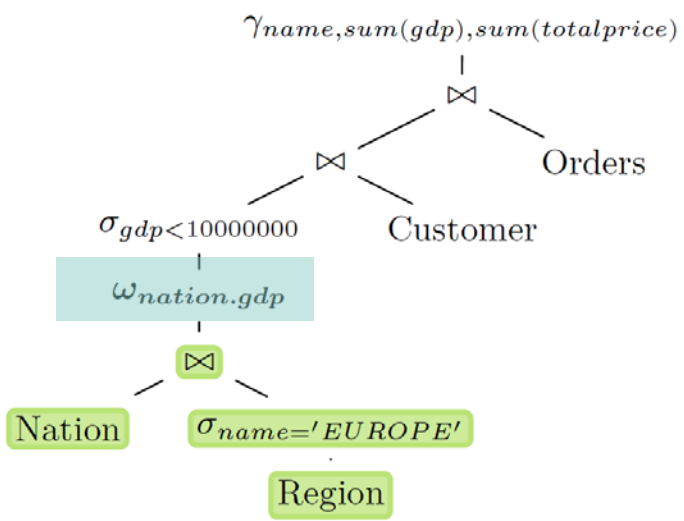
Multi-solution Query Processing

- 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

Dresden Database Systems Group

INIT()

- Inititializes state

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

NEXT()

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

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

DrillBeyond Operator ω (2)

Dresden Database Systems Group

COLLECT()

- Pulls and stores all tuples
- Blocking

AUGMENT()

- Pass all entries from the EAS into a hashtable

function COLLECT
  while *true* **do**
    $t \leftarrow \text{NEXT}(childPlan)$
    if $t = NULL$ **then**
      break
    $tuplestore \leftarrow t$
    $augKey \leftarrow \text{TEXTATTRS}(t)$
    if $augKey \notin augMap$ **then**
      $augMap[augKey] \leftarrow \emptyset$

function AUGMENT
  $augReq \leftarrow (\forall k \in augMap \mid augMap[k] = \emptyset)$
  for all $augKey, [augValues...] \in \text{SEND}(augReq)$ **do**
    $augMap[augKey] \leftarrow [augValues...]$

DrillBeyond Operator ω (3)

Dresden Database Systems Group

PROJECT()

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

RESCAN()

- Called when subtrees have to be re-executed

NEXTVARIANT()

- Produces the multi-variant query results

function PROJECT
  $t \leftarrow \text{NEXT}(tuplestore)$
  if $t = NULL$ **then return** $NULL$
  $augKey \leftarrow \text{TEXTATTRS}(t)$
  $t[augAttr] = augMap[augKey][n]$ **return** t

function RESCAN
  $state \leftarrow$ '$collecting$'
  $tuplestore \leftarrow \emptyset$

function NEXTVARIANT
  $\text{RESCAN}(tuplestore)$
  $n \leftarrow n + 1$

Why Blocking?

Dresden Database Systems Group

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

country	gdp
Russia	?
UK	?
USA	?

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

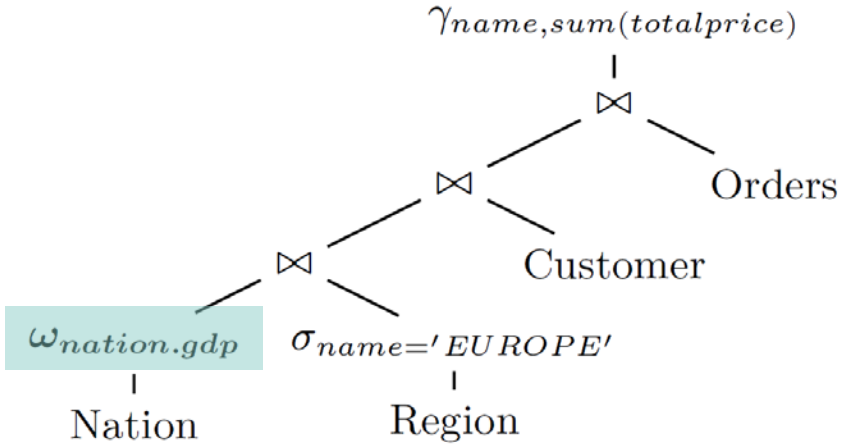
ds_4 European Economy

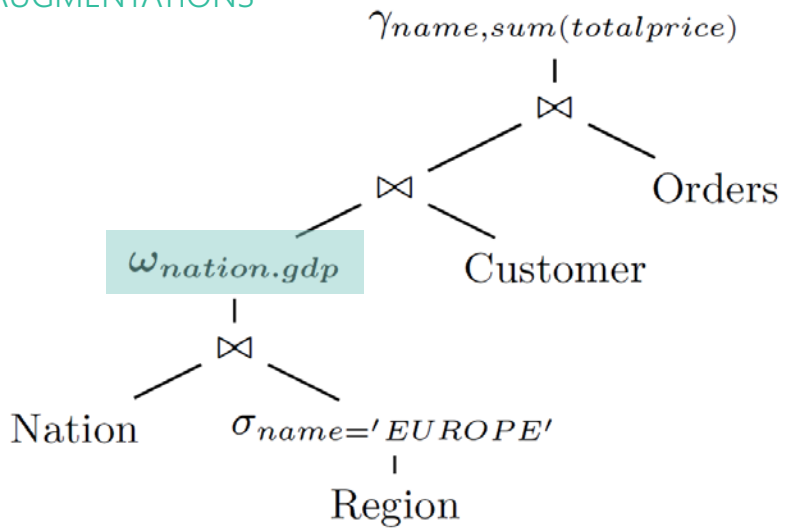
country	gdp EUR
UK	X

Lower Placement Bound

Dresden Database Systems Group

Same query, two different query plans → two different augmentations





Lower Placement Bound

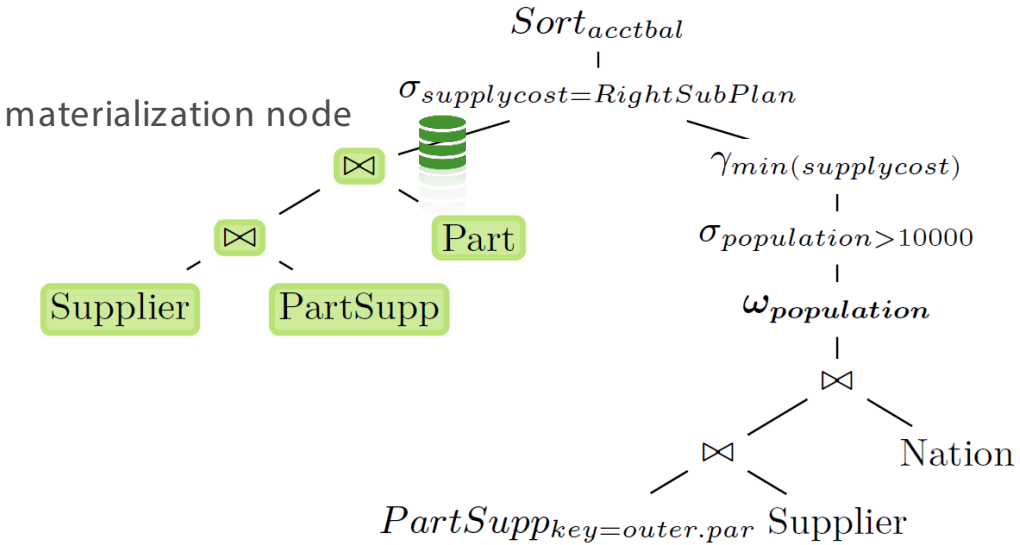
- ω can only be placed when all filters on R, e.g., joins and predicates, have been applied
- In other words: apply ω to the minimum number of entities in R

Invariant Caching

Dresden Database Systems Group

Example

- Modeled after TPC-H query 2



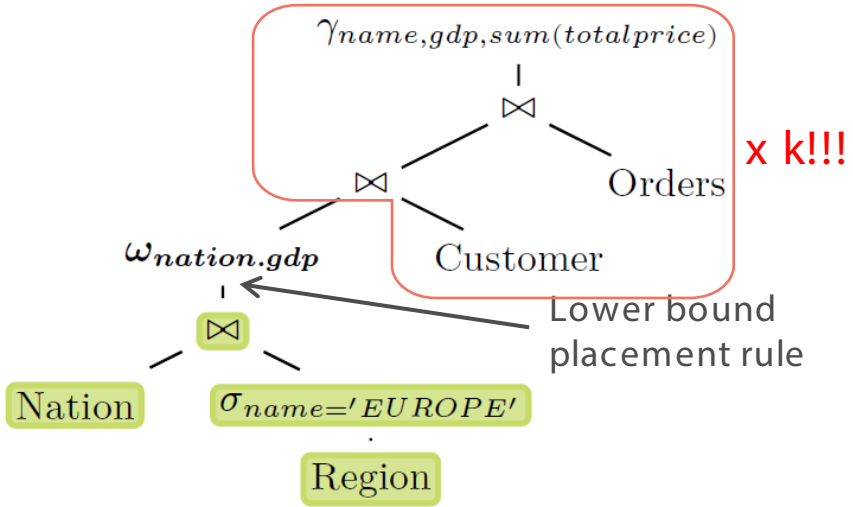
Limitation: depending on the cost-based placement of ω, most of the query may be varying

Maximize Invariant Sub-Trees

Dresden Database Systems Group

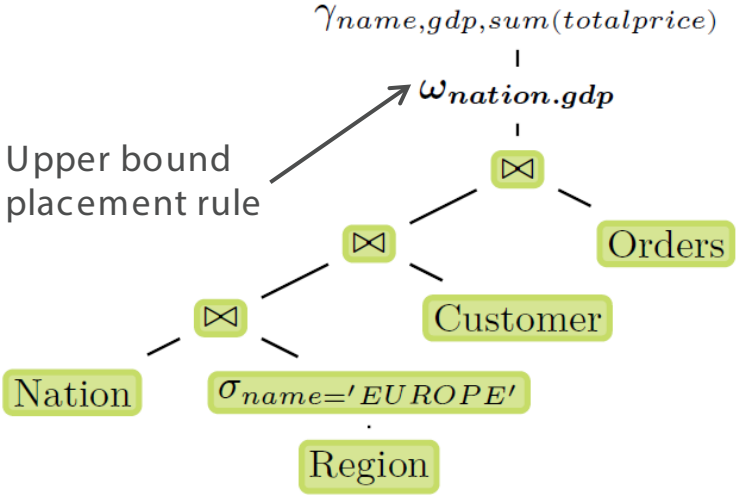
OPTIMAL PLAN...

- ..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

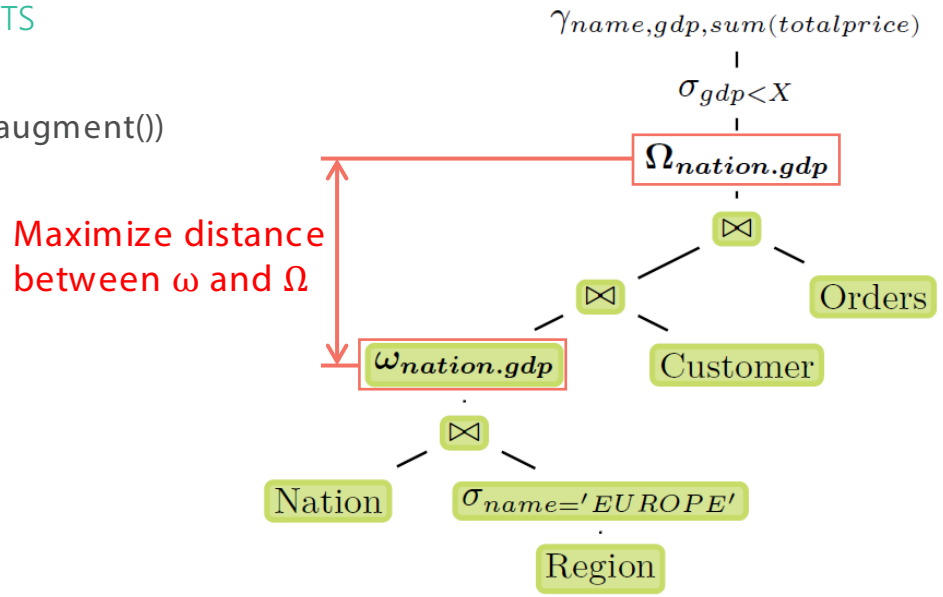
Observation

- 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

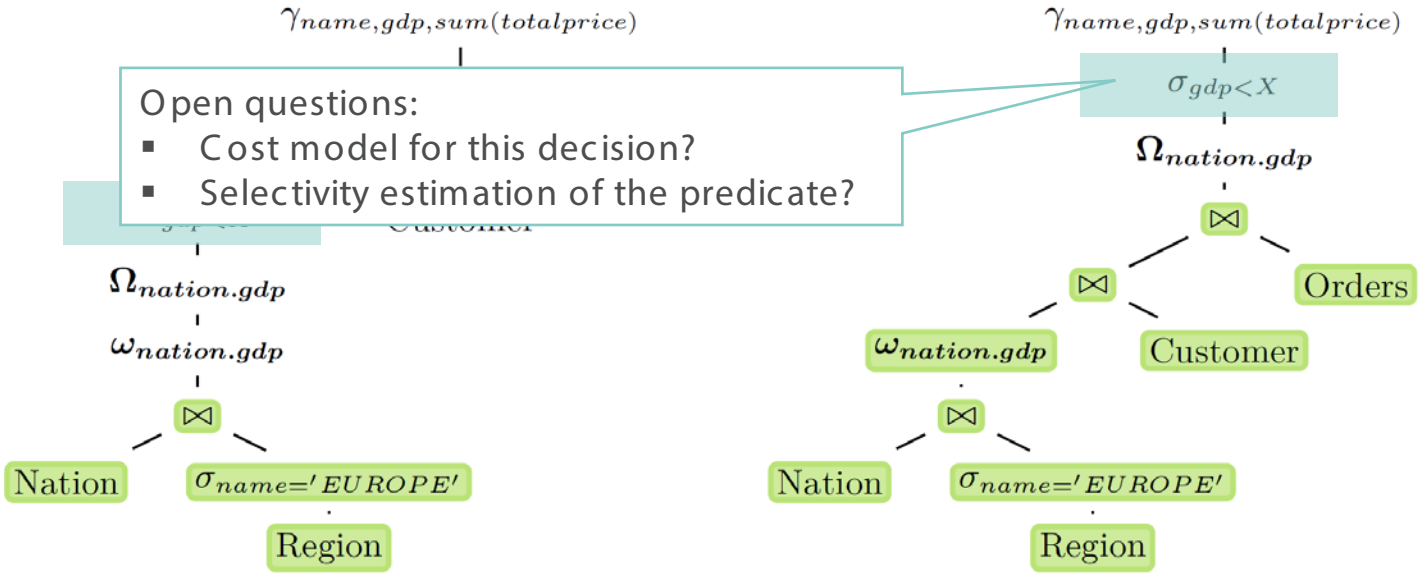
Dresden Database Systems Group

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

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

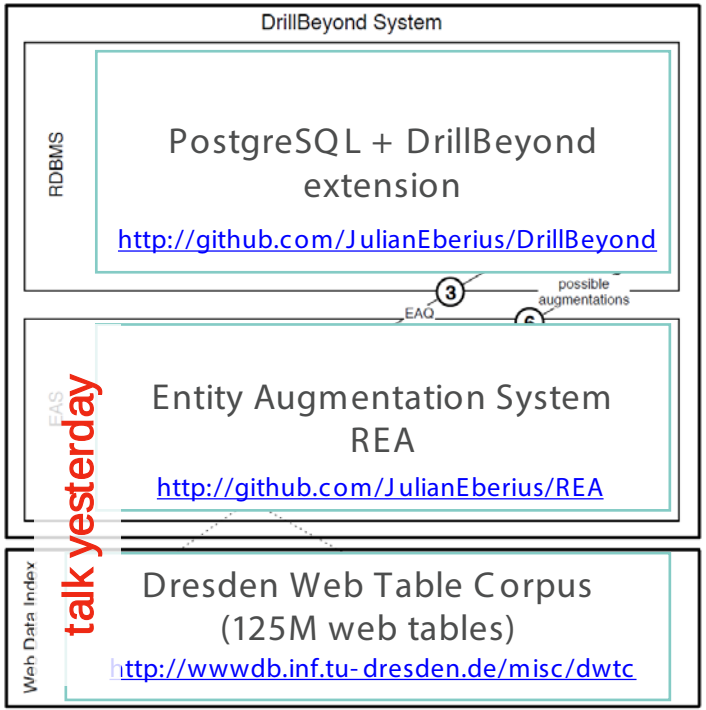
- $C(x,y)$: cost of the subplan from operator x to y
- s_i
  - Selectivity of the i[th] data source for the attribute in question
  - Not known at plan-time → determined in the augment() phase

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”

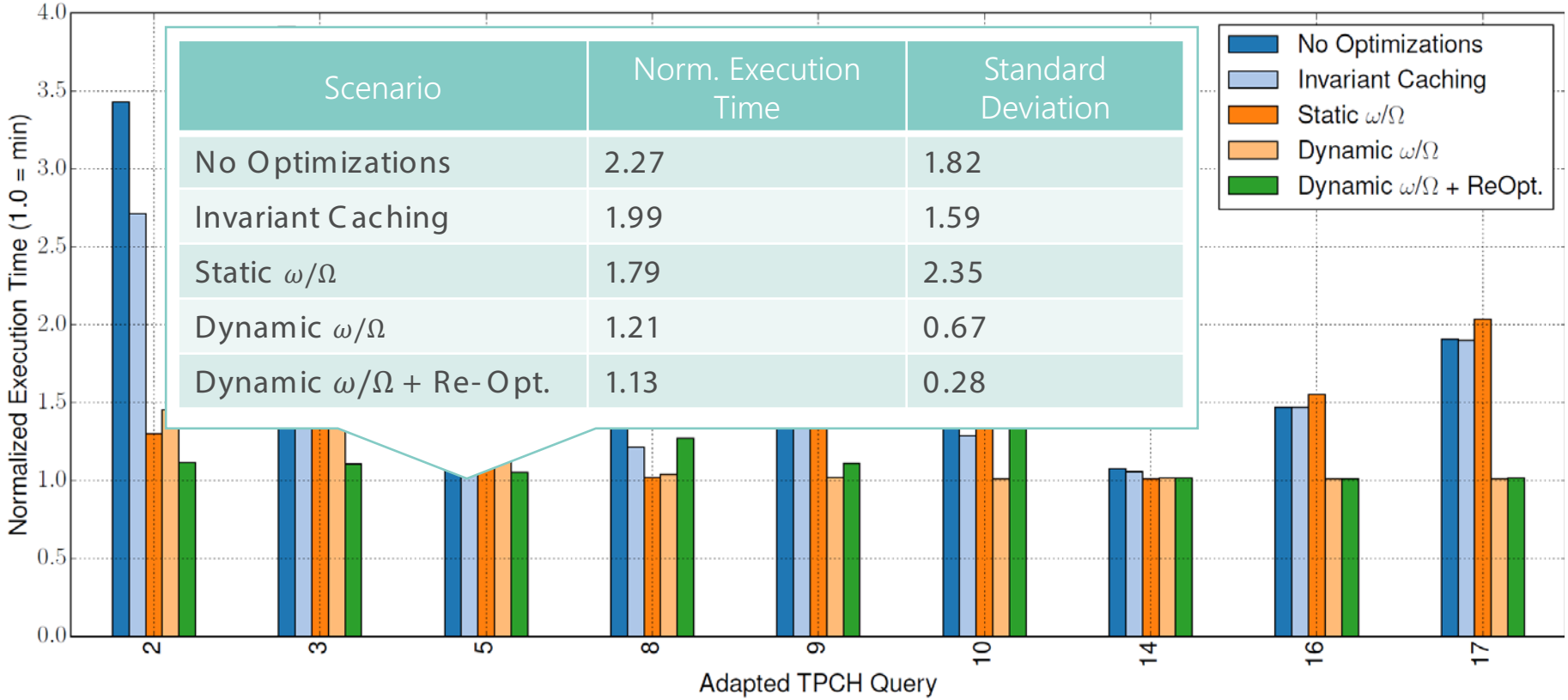
Parameters

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

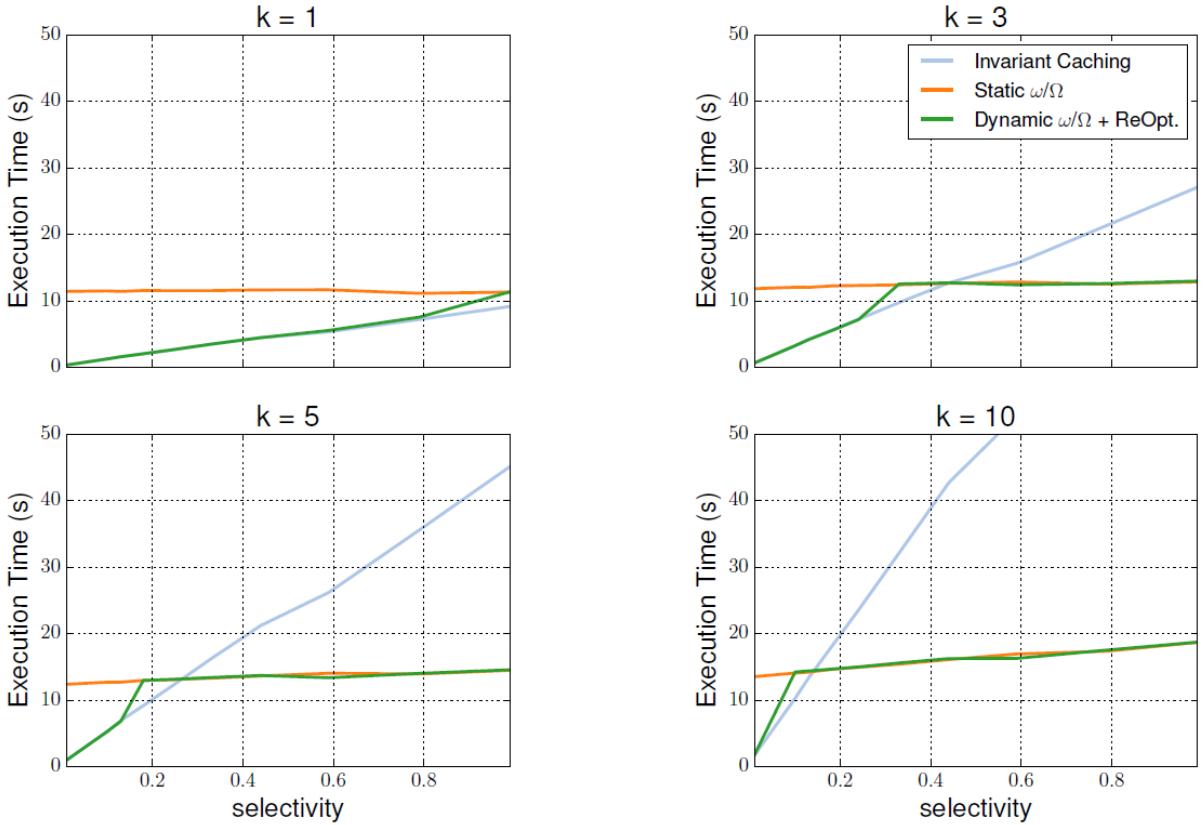
Overall Performance

Dresden Database Systems Group

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)