Towards Comprehensive Testing Tools

Redefining testing mechanisms!

PGCon 2017
26.05.2017

Kuntal Ghosh
(Software Engineer)

Contents

- Motivation
- Picasso Visualizer
- Picasso Art Gallery for PostgreSQL 10
- Plan Reduction
- CODD metadata processor

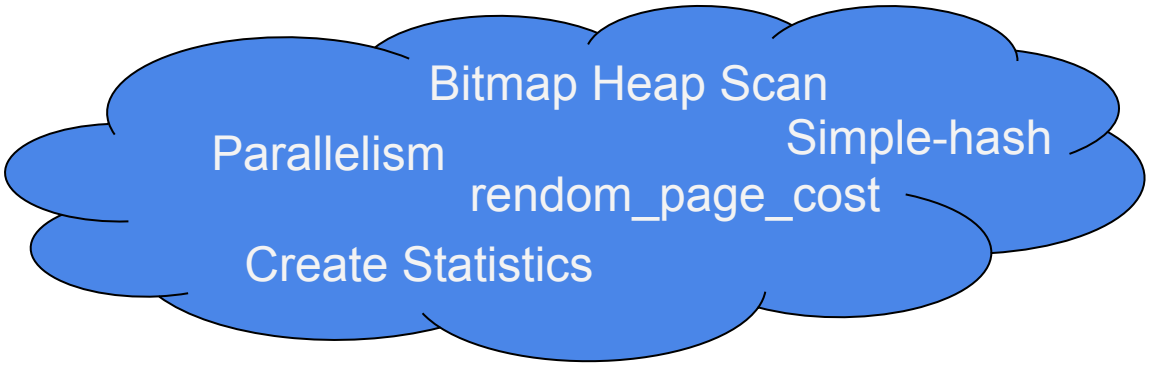
How can we improve query performance?

```
SELECT * FROM pgconf WHERE
num_of_attendees <= x AND price <= y
```

How can we improve query performance?

```
SELECT * FROM pgconf WHERE
num_of_attendees <= x AND price <= y
```

- Development of new operators
- Algorithm enhancement of existing operators
- Changes in optimizer's cost parameters or model



What should we test?

```
SELECT * FROM pgconf WHERE
num_of_attendees <= x AND price <= y
```

- What is the behavior of our enhancement for different values of x and y?
- Is the enhancement scalable?
- Is there any effect of the underlying data distribution?
- What happens if we change the cost parameters?

How should we test?

```
SELECT * FROM pgconf WHERE
num_of_attendees <= x AND price <= y
```

Optimizer-time testing

- involves creation of optimal execution plans by optimizer
  - metadata configurations are used by the query optimizer

How should we test?

```
SELECT * FROM pgconf WHERE
num_of_attendees <= x AND price <= y
```

Execution-time testing

- involves the execution of optimizer chosen plans
  - impractical(time) to test all possible query instances, e.g. for all x,y
  - infeasible(space) to explicitly create and/or process test data for alternate test scenarios, e.g. in a 100TB database

How should we test?

```
SELECT * FROM pgconf WHERE
num_of_attendees <= x AND price <= y
```

But, major decisions can be taken during query optimization by the optimizer

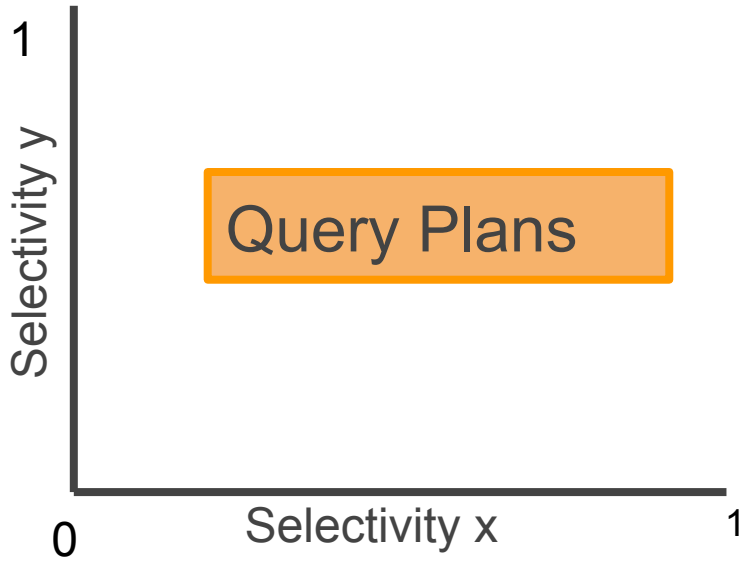
- Correctness
- Functional dependencies
- Operator choices
- Scalability

Hence, we are only interested in optimizer-time testing

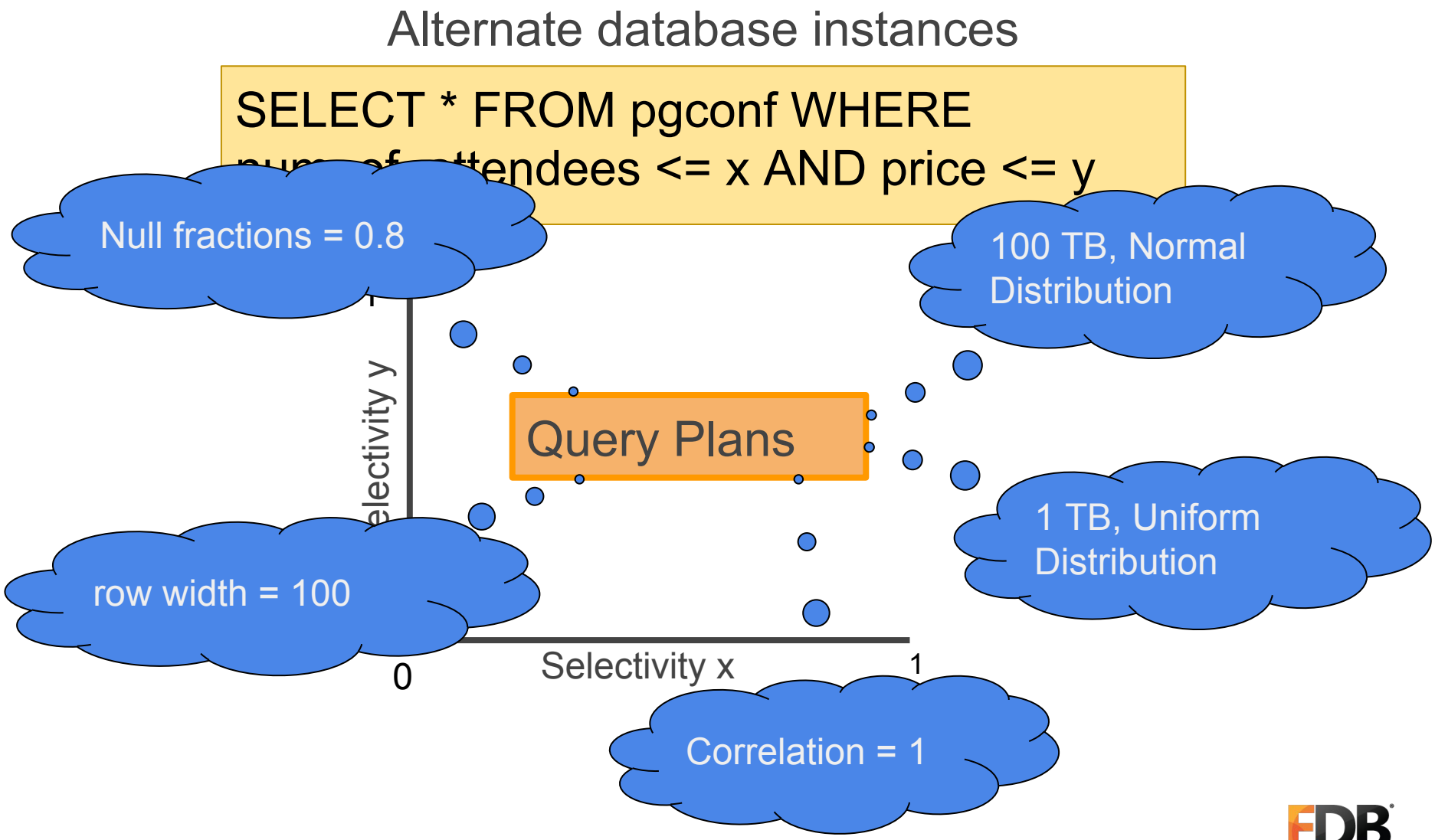
Optimizer-time testing

For a single database instance

```
SELECT * FROM pgconf WHERE
num_of_attendees <= x AND price <= y
```

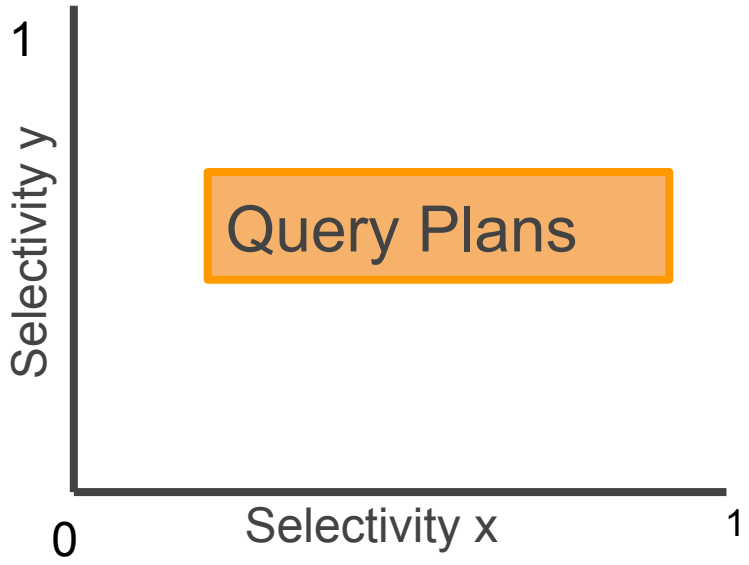


Optimizer-time testing



Tools at your rescue!

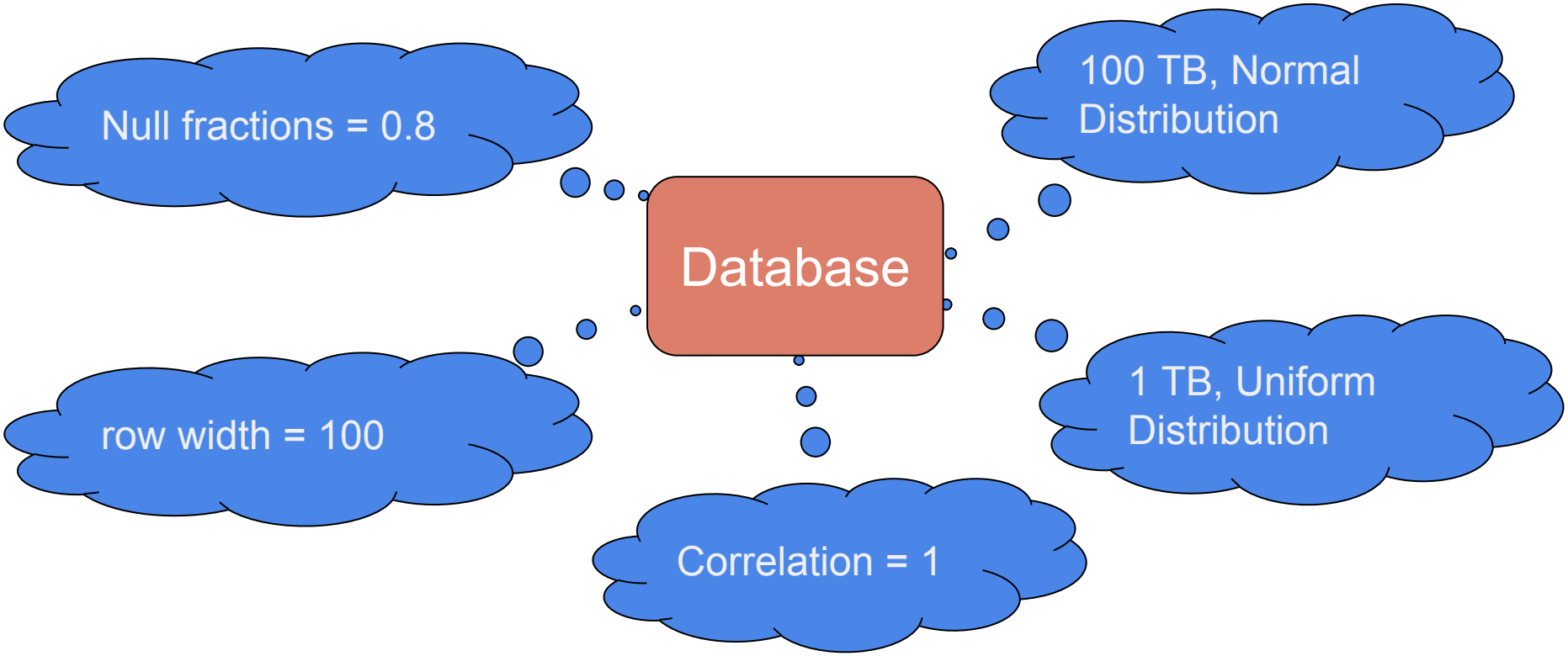
Given an n-dimensional selectivity space and a choice of database, Picasso* automatically generates query plans for a query template



*http://dsl.cds.iisc.ac.in/projects/PICASSO/

Tools at your rescue!

CODD* generates alternate test scenarios through "COnstruction of Dataless Database" in no time



*http://dsl.cds.iisc.ac.in/projects/CODD/

Contents

- Motivation
- Picasso Visualizer
- Picasso Art Gallery for PostgreSQL 10
- Plan Reduction
- CODD metadata processor

Picasso Visualizer*

Picasso is a (free) stand-alone Java tool that, given an n-dimensional SQL query template and a choice of database engine, automatically generates plan and cost diagrams

- Operational on
  - PostgreSQL
  - DB2
  - Oracle
  - SQLServer
  - Sybase
  - MySQL

Developed by Prof. Jayant R. Haritsa and his team at DSL, Indian Institute of Science, India

*http://dsl.cds.iisc.ac.in/projects/PICASSO/

Picasso Visualizer

- A plan diagram is a pictorial enumeration of the plan choices of the query optimizer over the relational selectivity space

- A cost diagram is a visualization of the (estimated) plan execution costs over the same relational selectivity space

TPC-H Q7 Template

TPC-H Q7 query template

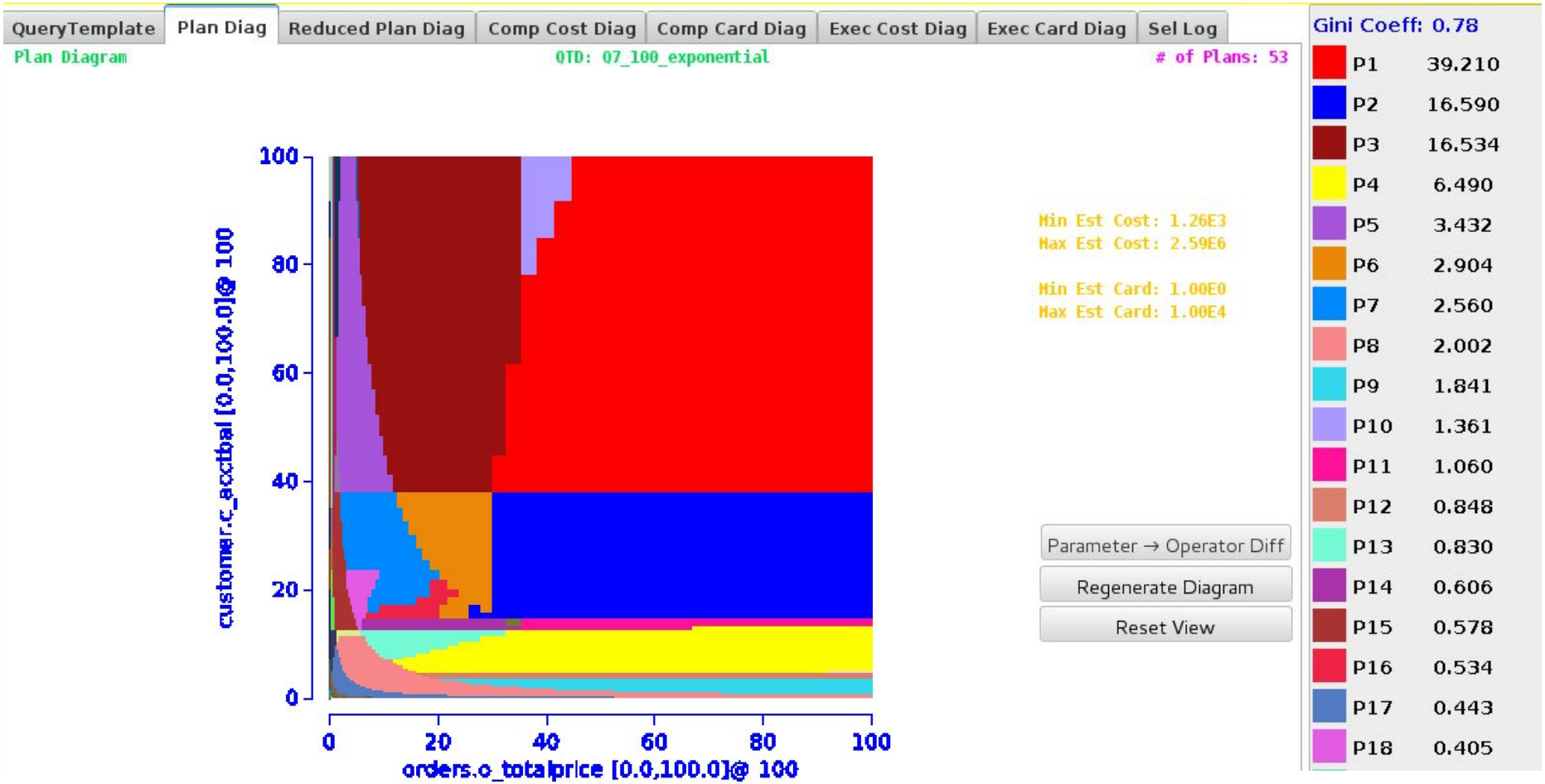
select supp_nation, cust_nation, l_year, sum(volume) as revenue from (select n1.n_name as supp_nation, n2.n_name as cust_nation, extract(year from l_shipdate) as l_year, l_extendedprice * (1 - l_discount) as volume from supplier, lineitem, orders, customer, nation n1, nation n2 where s_suppkey = l_suppkey and o_orderkey = l_orderkey and c_custkey = o_cust

Value determines selectivity of Customer Relation

Value determines selectivity of Orders Relation

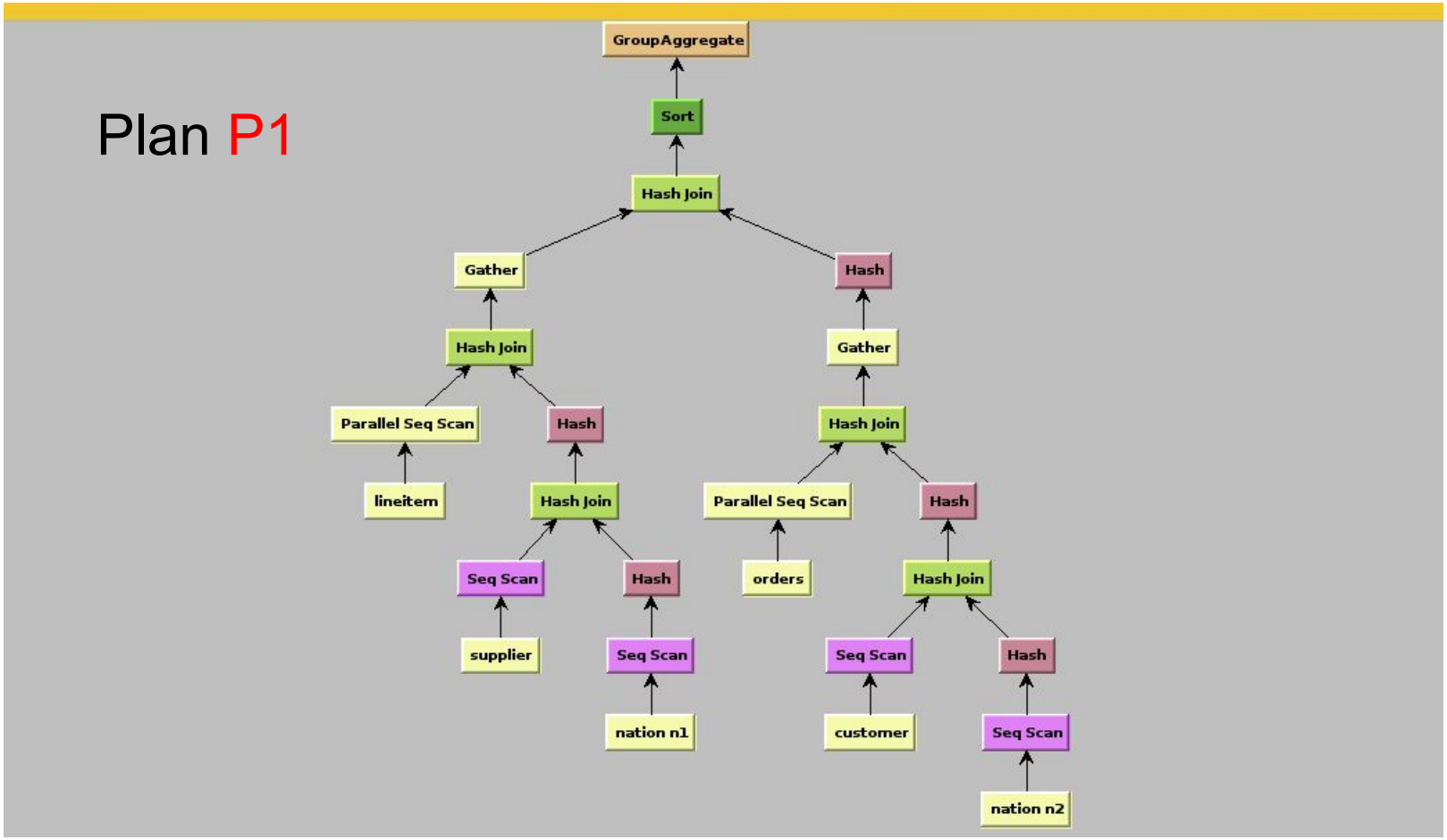
= 'FRANCE')) and pdate between date '1995-01-01' and date '1996-12-31' and o_totalprice <=c1 and c_acctbal <=c2) as shipping group by supp_nation, cust_nation, l_year order by supp_nation, cust_nation, l_year

Picasso Visualizer for TPC-H Q7

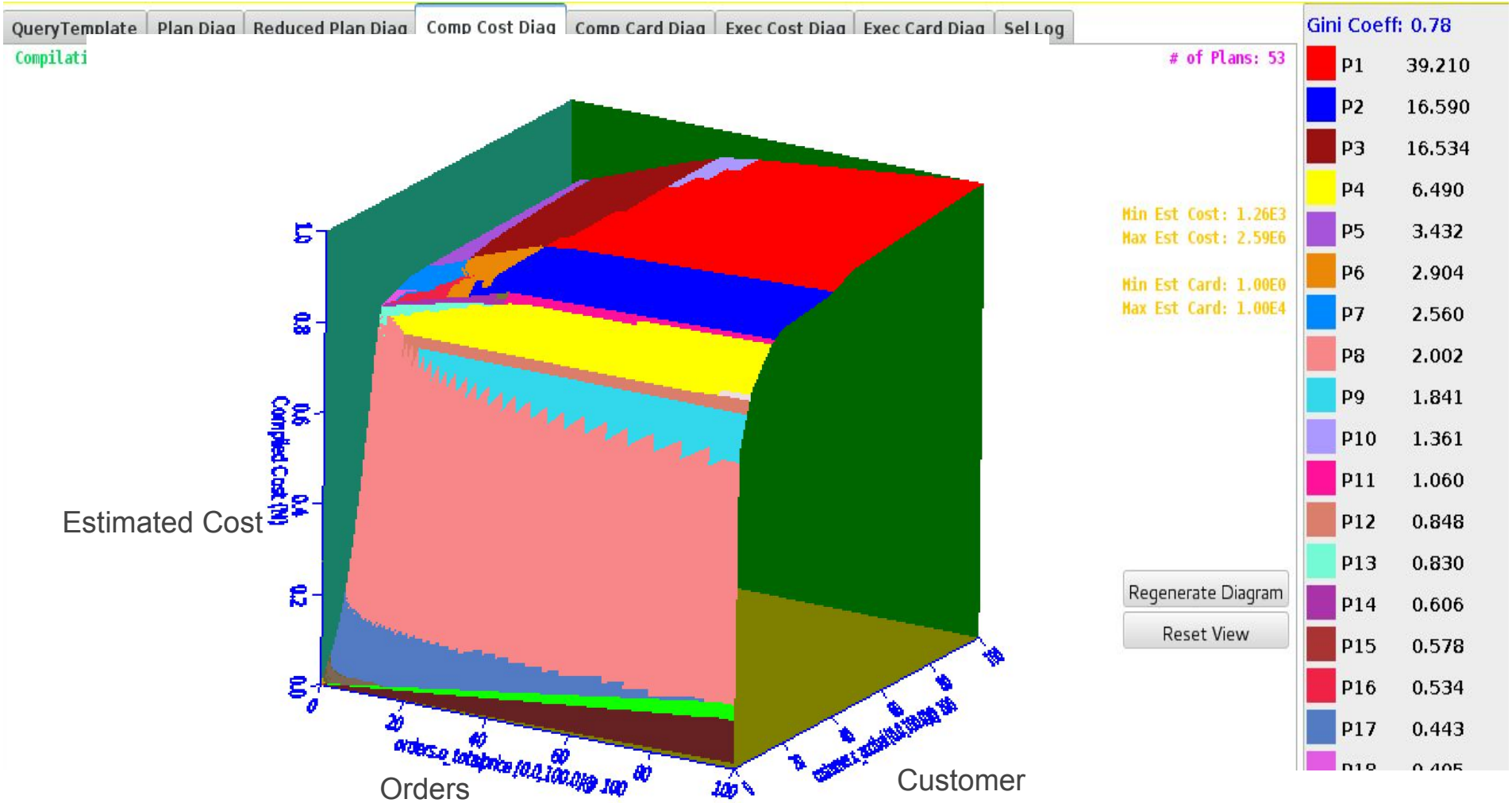


Plan Diagram

Picasso Visualizer for TPC-H Q7

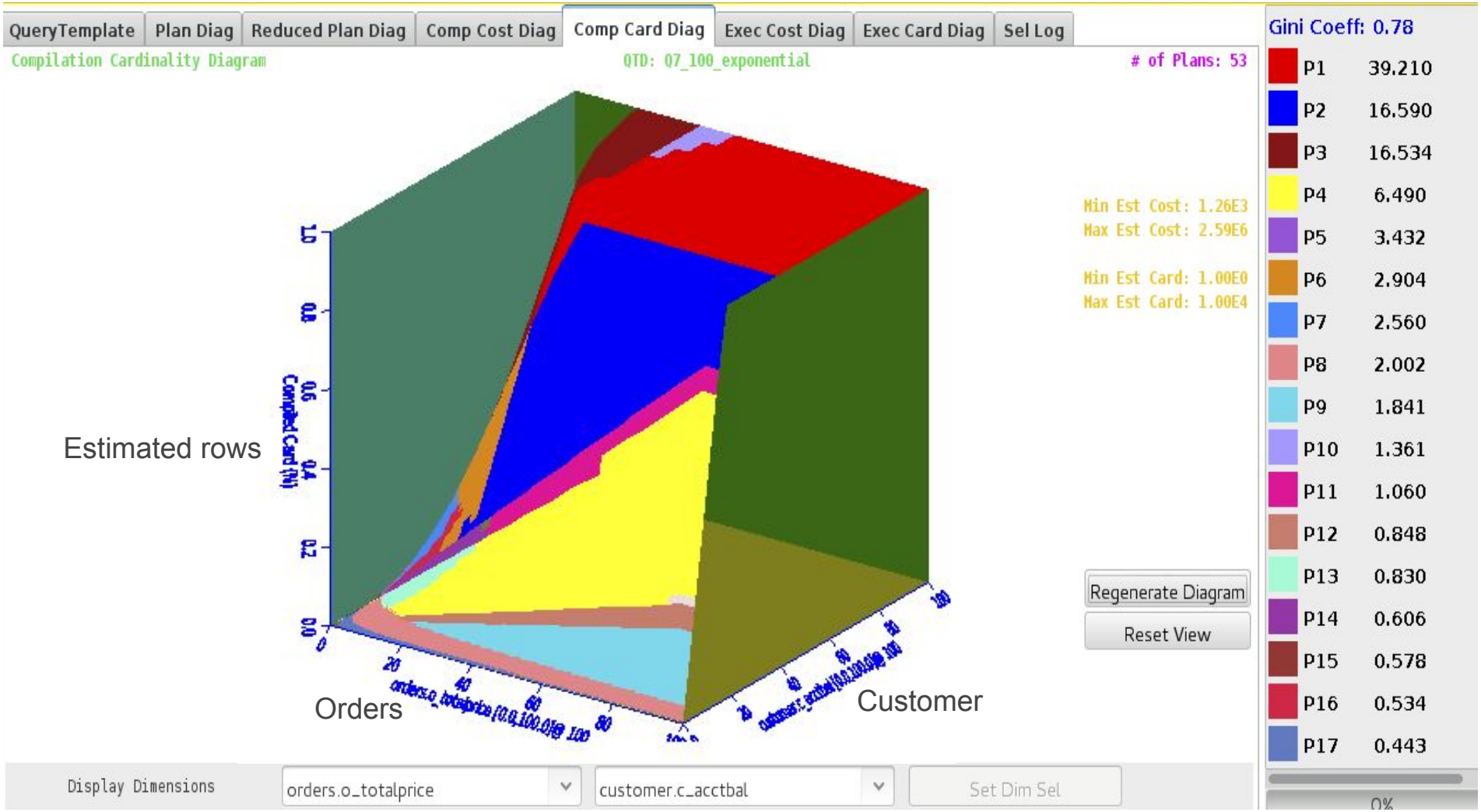


Picasso Visualizer for TPC-H Q7



Cost Diagram

Picasso Visualizer for TPC-H Q7



Cardinality Diagram

Picasso Connection

Woman with a guitar
Georges Braque, 1913

[Pablo Picasso founder
of cubist genre]

Plan diagrams are often similar to cubist paintings !

Contents

- Motivation
- Picasso Visualizer
- Picasso Art Gallery for PostgreSQL 10
- Plan Reduction
- CODD metadata processor

Picasso Art Gallery for PostgreSQL 10

Basic tenets of Parametric Query Optimization (PQO) literature

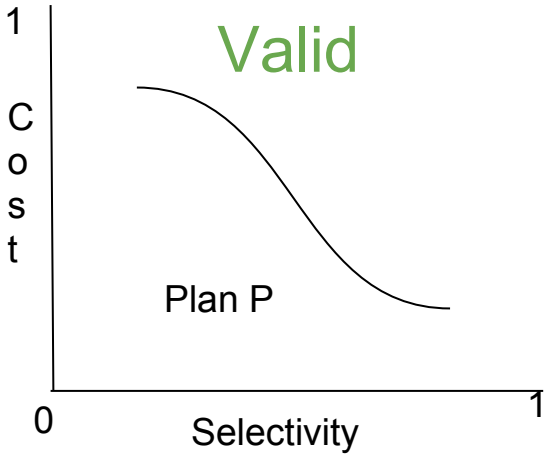
- Plan Convexity: Plan optimal at X and Y, is also optimal at all locations on the line joining X and Y
- Plan Uniqueness: An optimal plan appears at only a single contiguous region in the space
- Plan Homogeneity: An optimal plan is optimal within the entire region enclosed by its boundaries

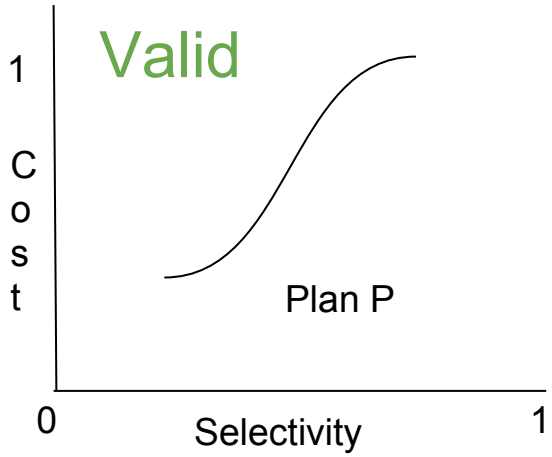
Picasso Art Gallery for PostgreSQL 10

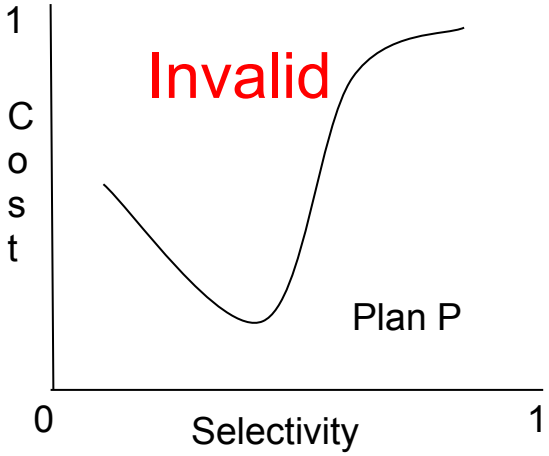
Basic tenets of Parametric Query Optimization (PQO) literature

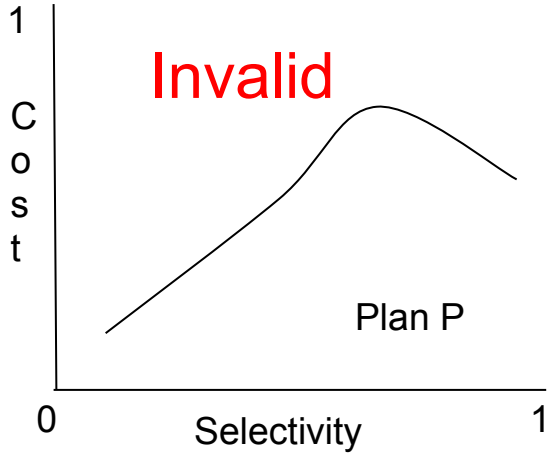
- For a single plan, Cost changes should be monotonic in behaviour(increase or decrease)

Picasso Art Gallery for PostgreSQL 10



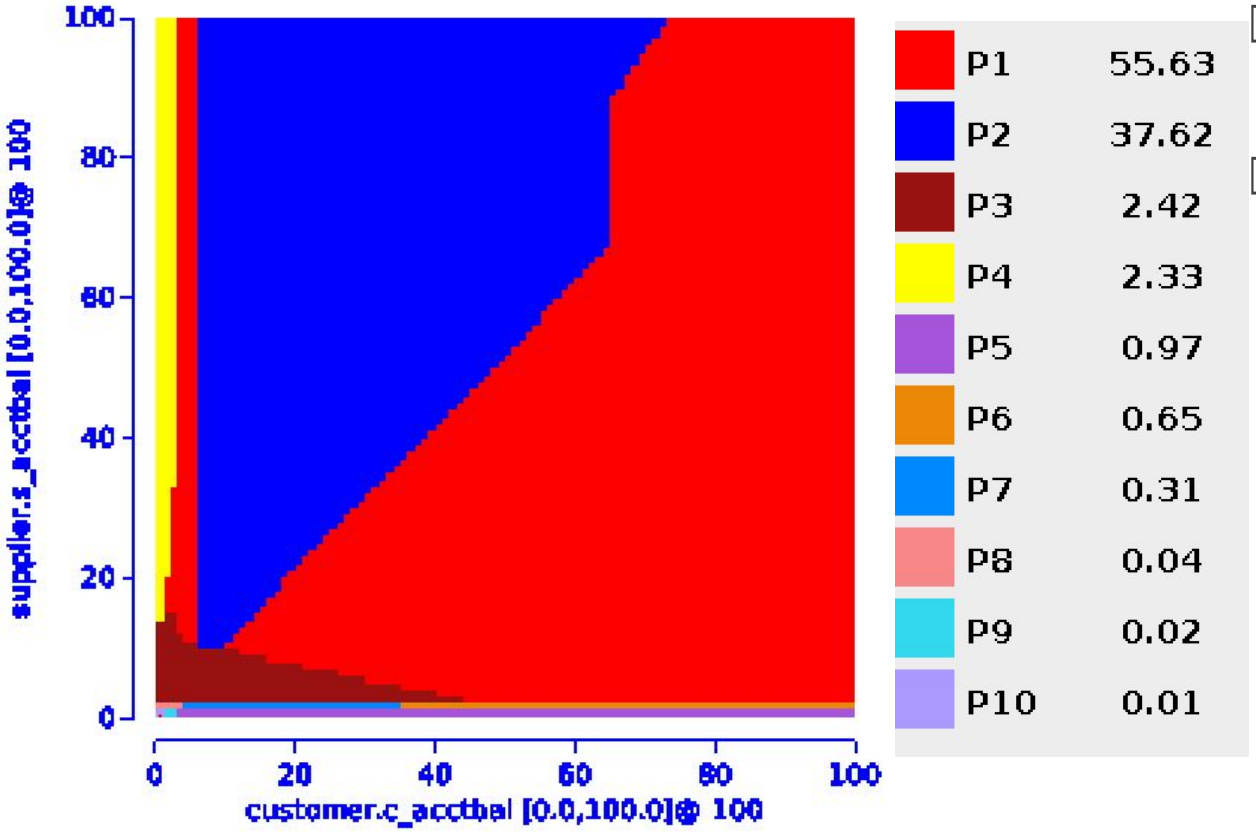






Picasso Art Gallery for PostgreSQL 10

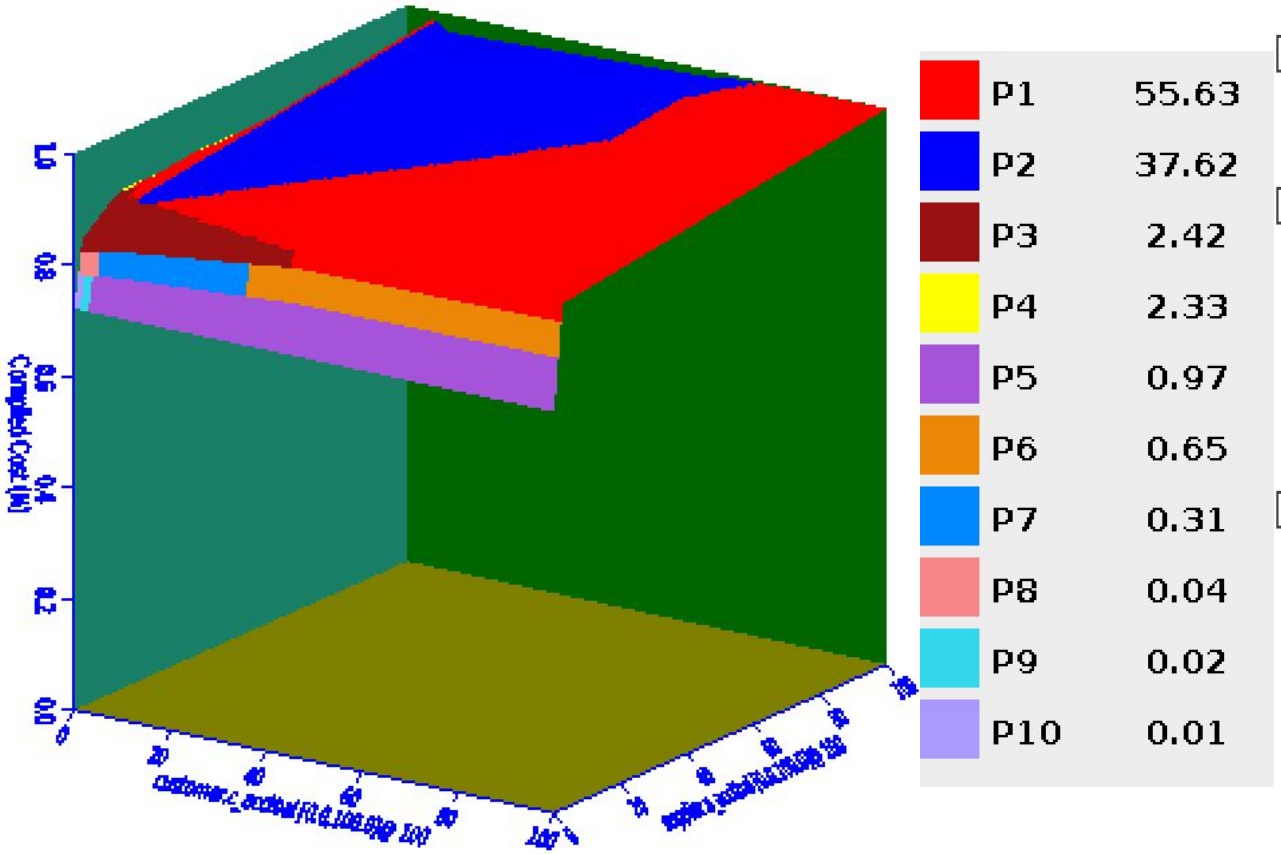
TPC-H Q5, Scale Factor 10



- P1 appearing at two places.
- Only difference between P1 and P2 is change in Hash join order

Picasso Art Gallery for PostgreSQL 10

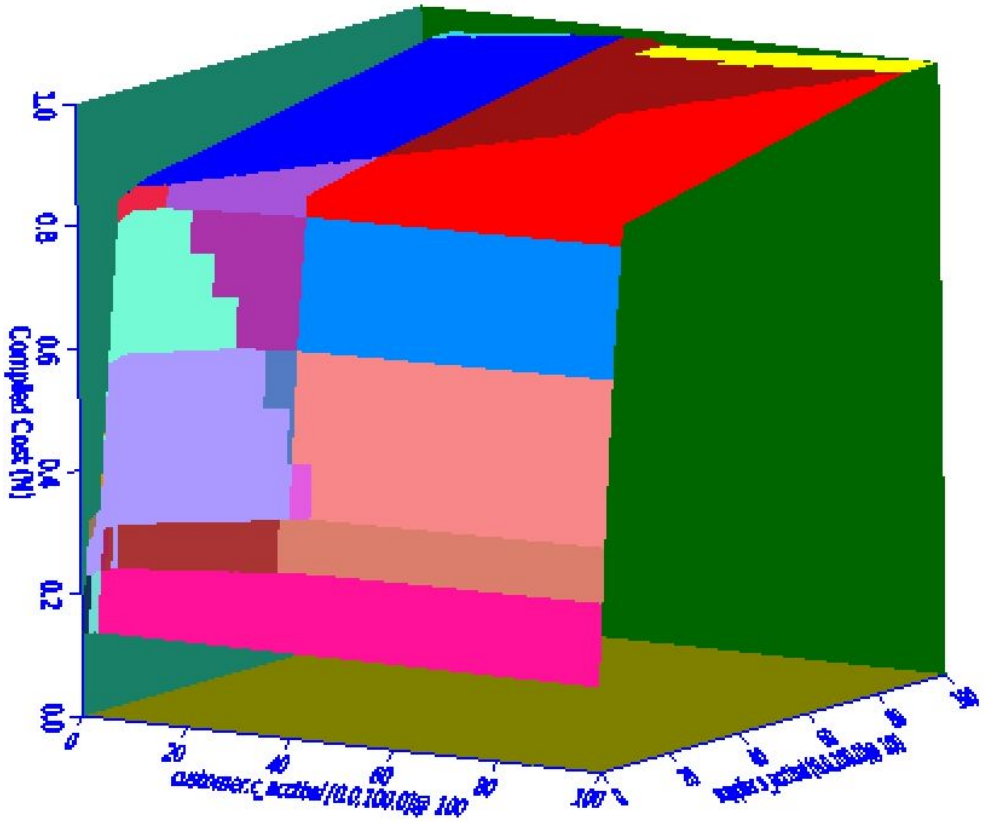
TPC-H Q5, Scale Factor 10



- P1 appearing at two places.
- Only difference between P1 and P2 is change in Hash join order
- Cost difference is not significant between P1 and P2

Picasso Art Gallery for PostgreSQL 10

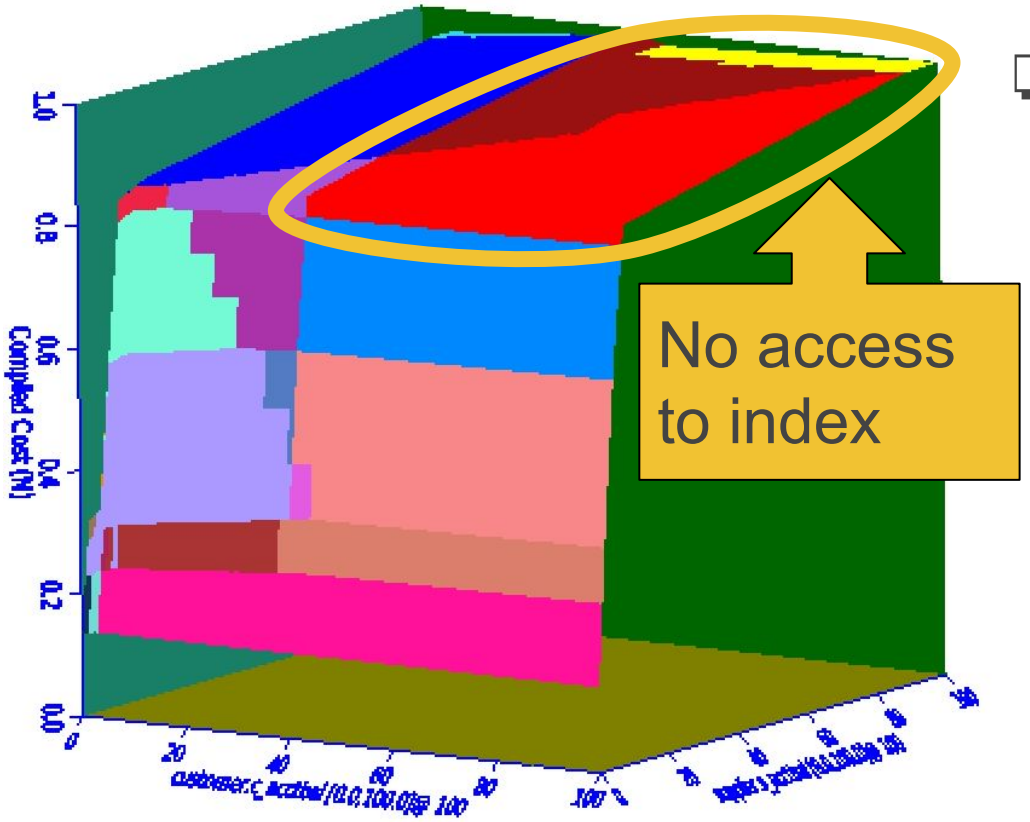
TPC-H Q5, Scale Factor 10 with indexes on all columns



How does the cost change with additional index creation?

Picasso Art Gallery for PostgreSQL 10

TPC-H Q5, Scale Factor 10 with indexes on all columns



- Indexes are not useful for most of the region in the selectivity space apart from regions near to axis

Observations

- The newly added features like Gather-merge, Parallel-bitmap scan, Parallel Index scans are used in significant regions of plan diagrams generated by TPC-H queries with scale factor 10 and they have reduced the cost
- Increasing the scale factor may increase the use of parallel index-only scans
- No significant cost non-monotonicity is observed

Contents

- Motivation
- Picasso Visualizer
- Picasso Art Gallery for PostgreSQL 10
- Plan Reduction
- CODD metadata processor

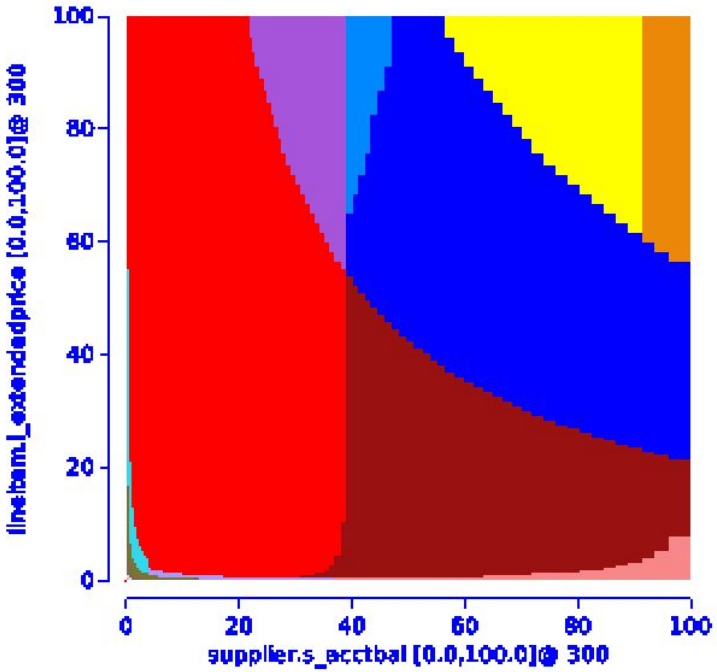
Plan Reduction

Can the plan diagram be recolored with a smaller set of colors (i.e. some plans are “swallowed” by others), such that Guarantee:

No query point in the original diagram has its estimated cost increased, post-swallowing, by more than λ percent (user-defined)

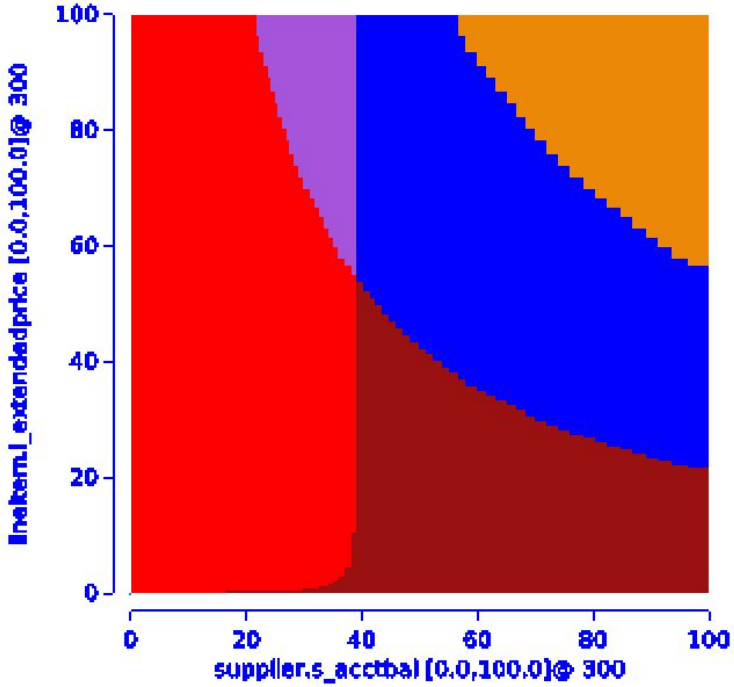
Plan Reduction

TPC-H Q8, Scale Factor 10



#Plans: 46

λ = 10%



#Plans: 6

Plan Reduction

Applications

- Retained plans are robust choices over larger regions of the selectivity space
- Provides better candidates for plan-caching

Contents

- Motivation
- Picasso Visualizer
- Picasso Art Gallery for PostgreSQL 10
- Plan Reduction
- CODD metadata processor

CODD metadata processor*

- Wish to test "Yottabyte" (10^{24}) scale database environment
- Impractical(time) and/or infeasible(space) to explicitly create and/or process test data
- A testing environment, wherein the entire data and metadata is virtual/transient, supporting efficient evaluation of alternate test scenarios

CODD metadata processor*

- CODD, a (free)graphical tool, “fools” the underlying system into thinking that data is actually present even though it’s never been created/stored

- Developed by Prof. Jayant R. Haritsa and his team at DSL, Indian Institute of Science, India

*http://dsl.cds.iisc.ac.in/projects/CODD/

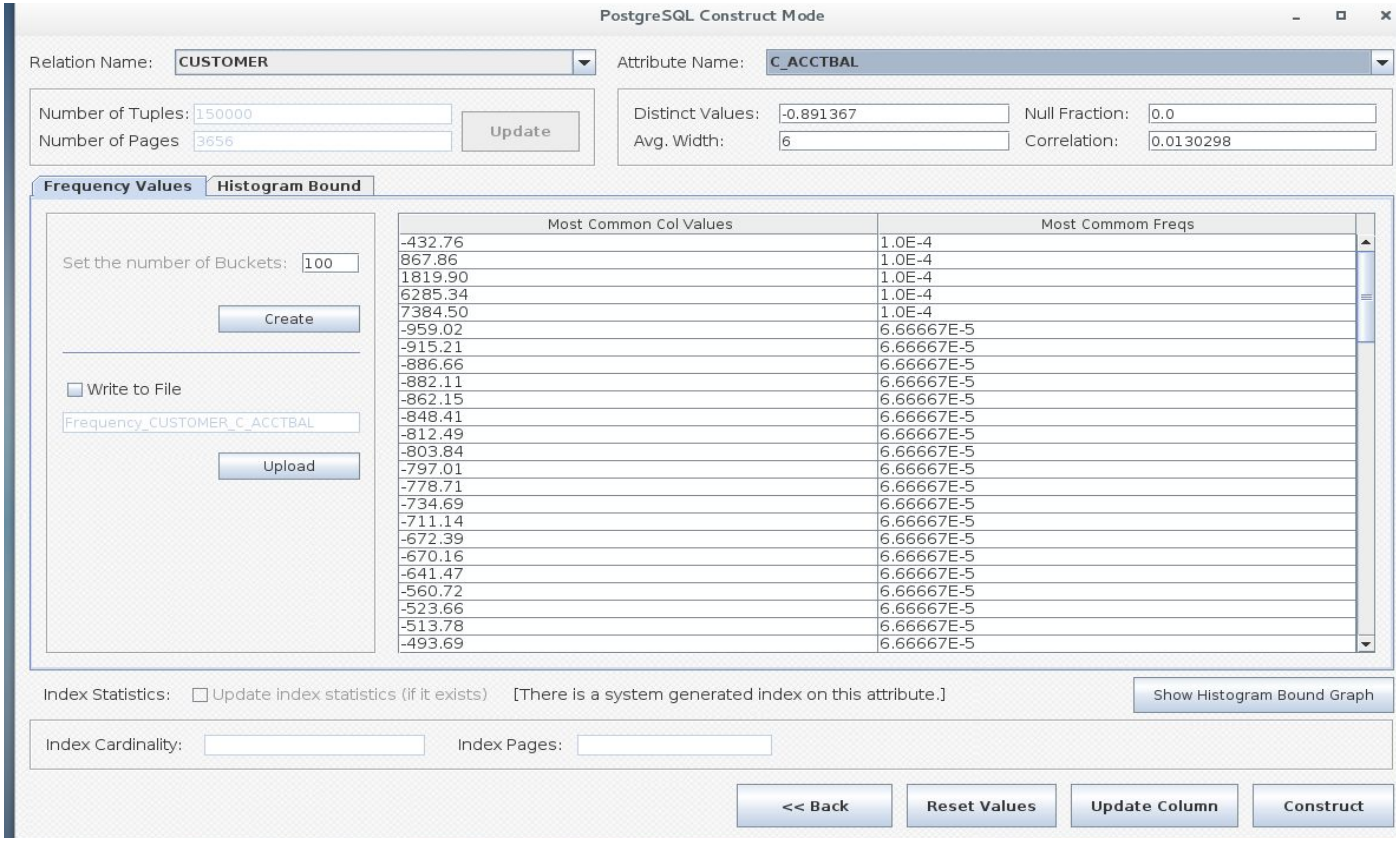
CODD metadata processor

Metadata testing with CODD

- Automated Construction of metadata
- Metadata scaling
- Metadata retention
- Porting of relational metadata configurations

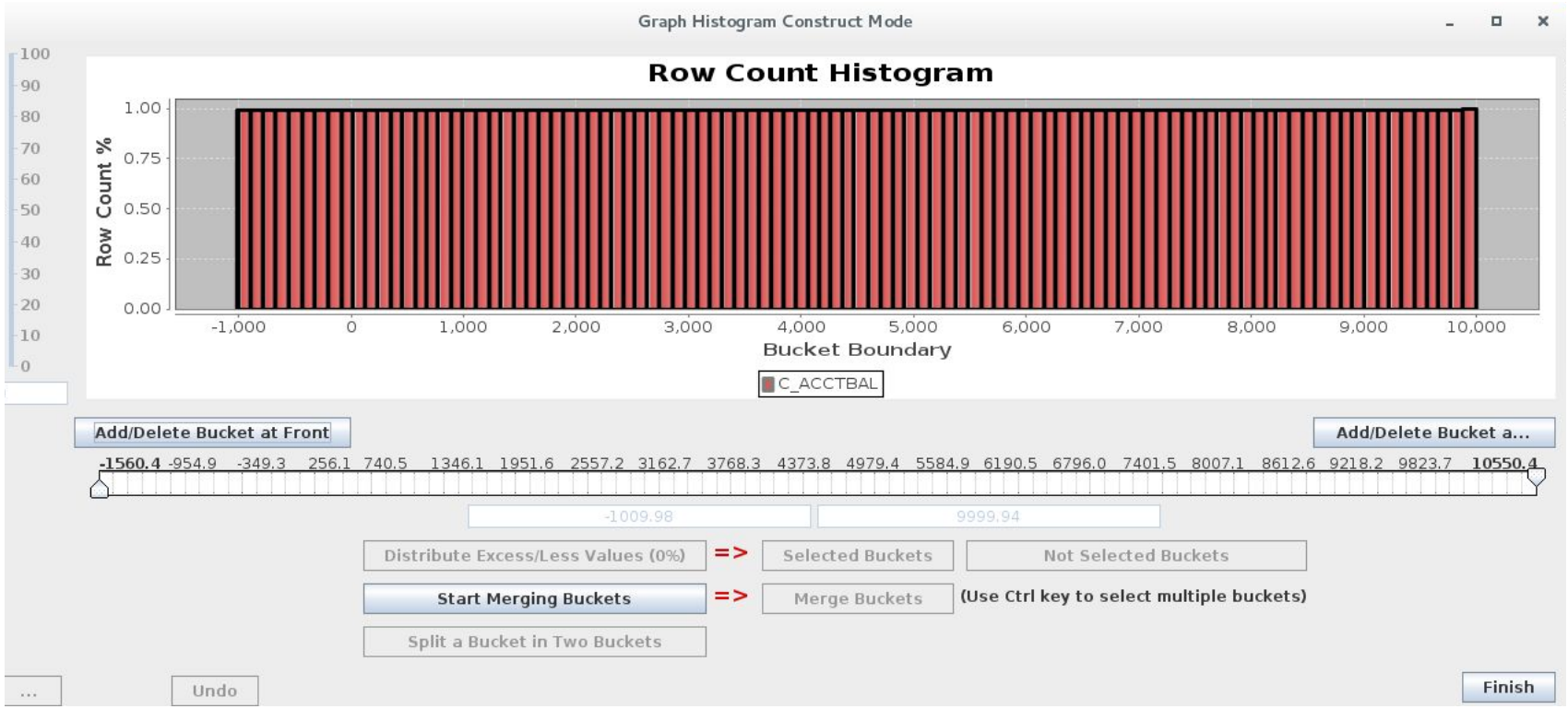
CODD metadata processor

Automated Construction of metadata (CREATE/MODIFY)



CODD metadata processor

Automated Construction of metadata (CREATE/MODIFY)



CODD metadata processor

Automated Construction of metadata (CREATE/MODIFY)

- CODD ensures that the inputted information is both
  - legal (valid type and range)
  - consistent (compatible with other metadata values)

It can be used for

- Testing optimizer behaviour in futuristic scenarios.
- Construction of metadata favourable for an operator

CODD metadata processor

Metadata scaling

- CODD supports size-based scaling for PostgreSQL
  - Linearly scales an initial metadata M by a positive scaling factor α, such that the size of the scaled database is α times the initial database size

It can be used for

- Testing scalability

CODD metadata processor

Metadata retention

- CODD allows the user to fully reclaim the storage currently occupied by the database instance without triggering any changes in the associated metadata

It can be used for

- Optimizer testing in low disk space

CODD metadata processor

Metadata porting

Engine	DB2	Oracle	SQL Server	Sysbase	PostgreSQL
DB2	Y	Y	N	Partial	Y
Oracle	Partial	Y	N	Partial	Partial
SQL Server	Y	Y	Y	Partial	Y
Sysbase	Y	Y	N	Y	Partial
PostgreSQL	Y	Y	N	Partial	Y

Cell(a,b): Metadata from column a can be ported to row b

CODD metadata processor

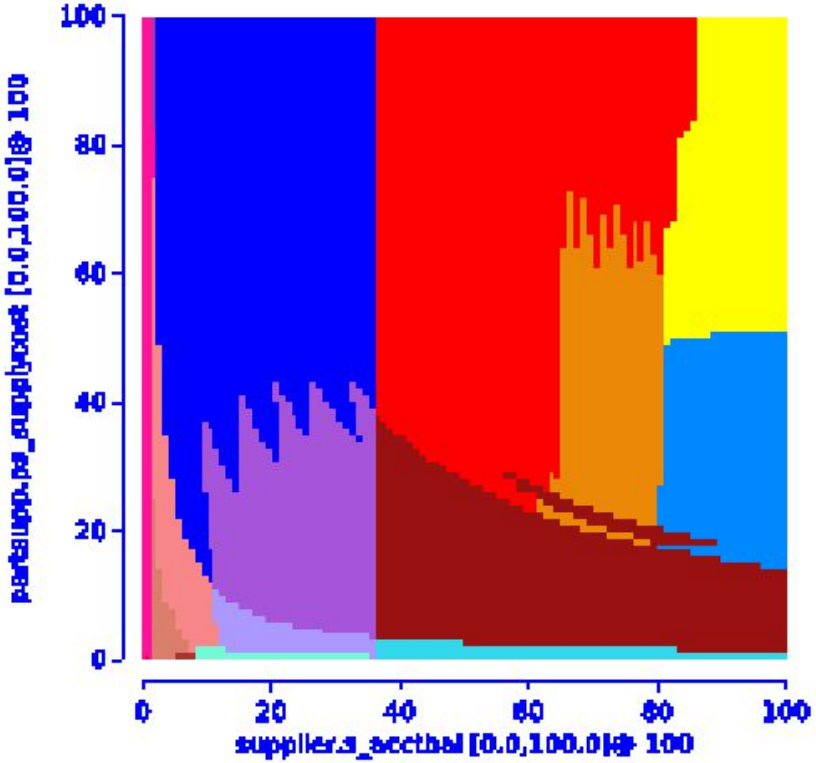
Metadata porting

It can be used for

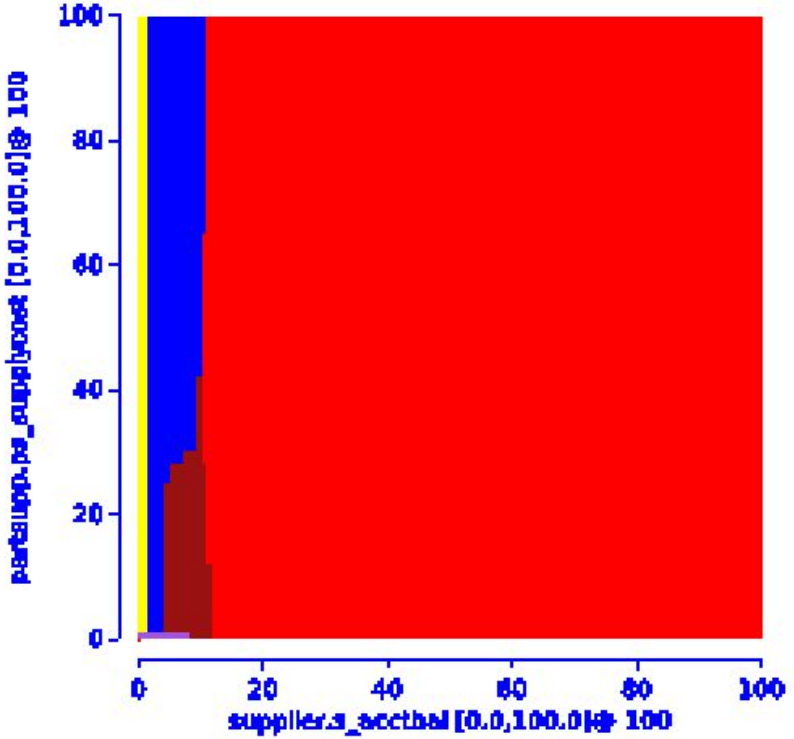
- Comparison of plan choices across different database engines for a given metadata
- Simulate production environment for query optimization testing

Utility of CODD

TPC-H Q9, 1GB



TPC-H Q9, 100TB



References

[1] http://dsl.cds.iisc.ac.in/projects/PICASSO/

[2] http://dsl.cds.iisc.ac.in/projects/PICASSO/picasso_download/icde_tutorial_updated.pdf

[3] http://dsl.cds.iisc.ac.in/projects/CODD/

[4] http://dsl.cds.iisc.ac.in/publications/report/TR/TR-2012-02.pdf

Thank You