Finding the Pitfalls in Query Performance

M.L. Kersten

P. Koutsourakis

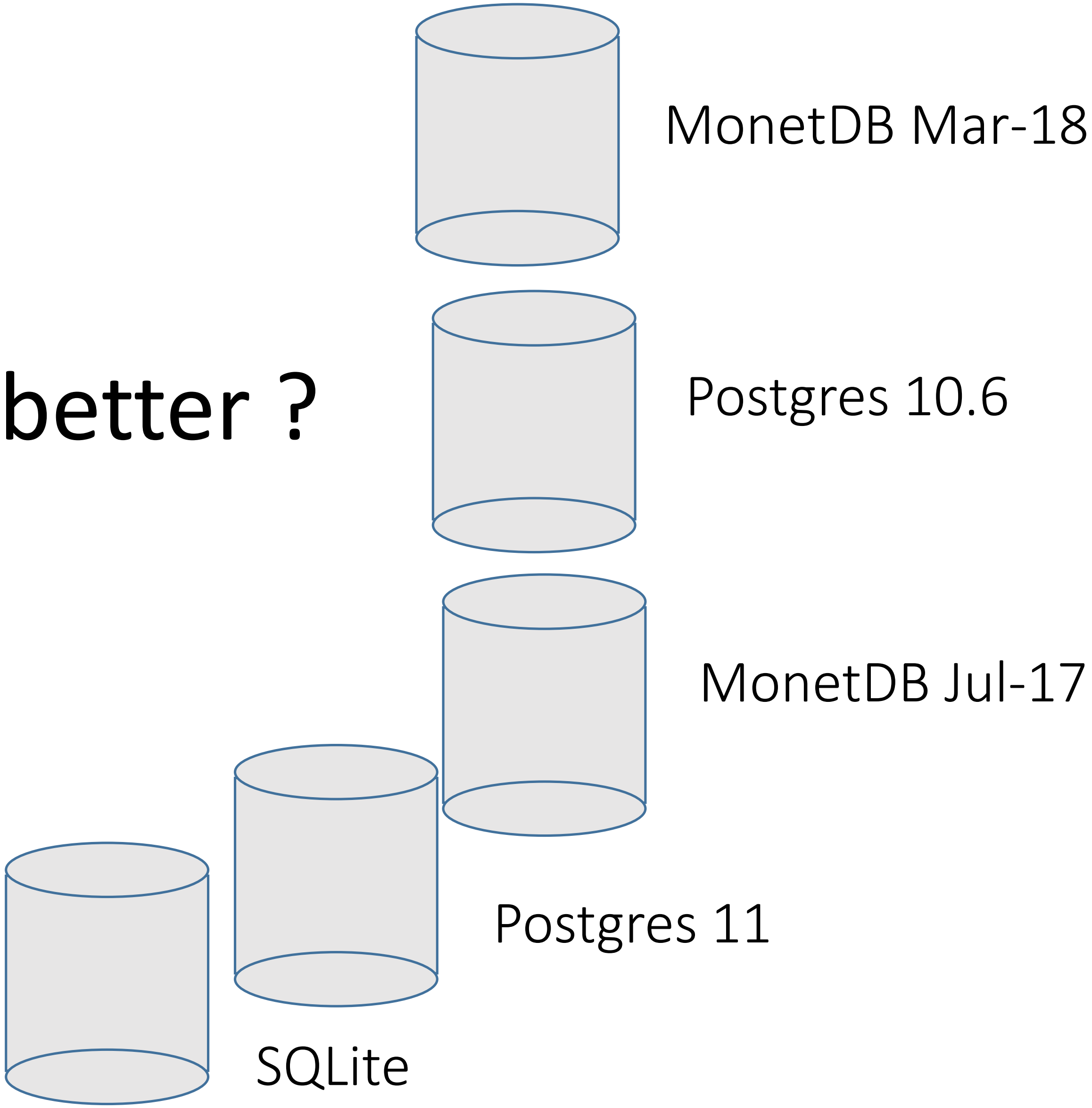
Y. Zhang

CWI, MonetDB Solutions

EU H2020 project ACTiCLOUD

The Challenge

Which system is relatively better ?



MonetDB Mar-18

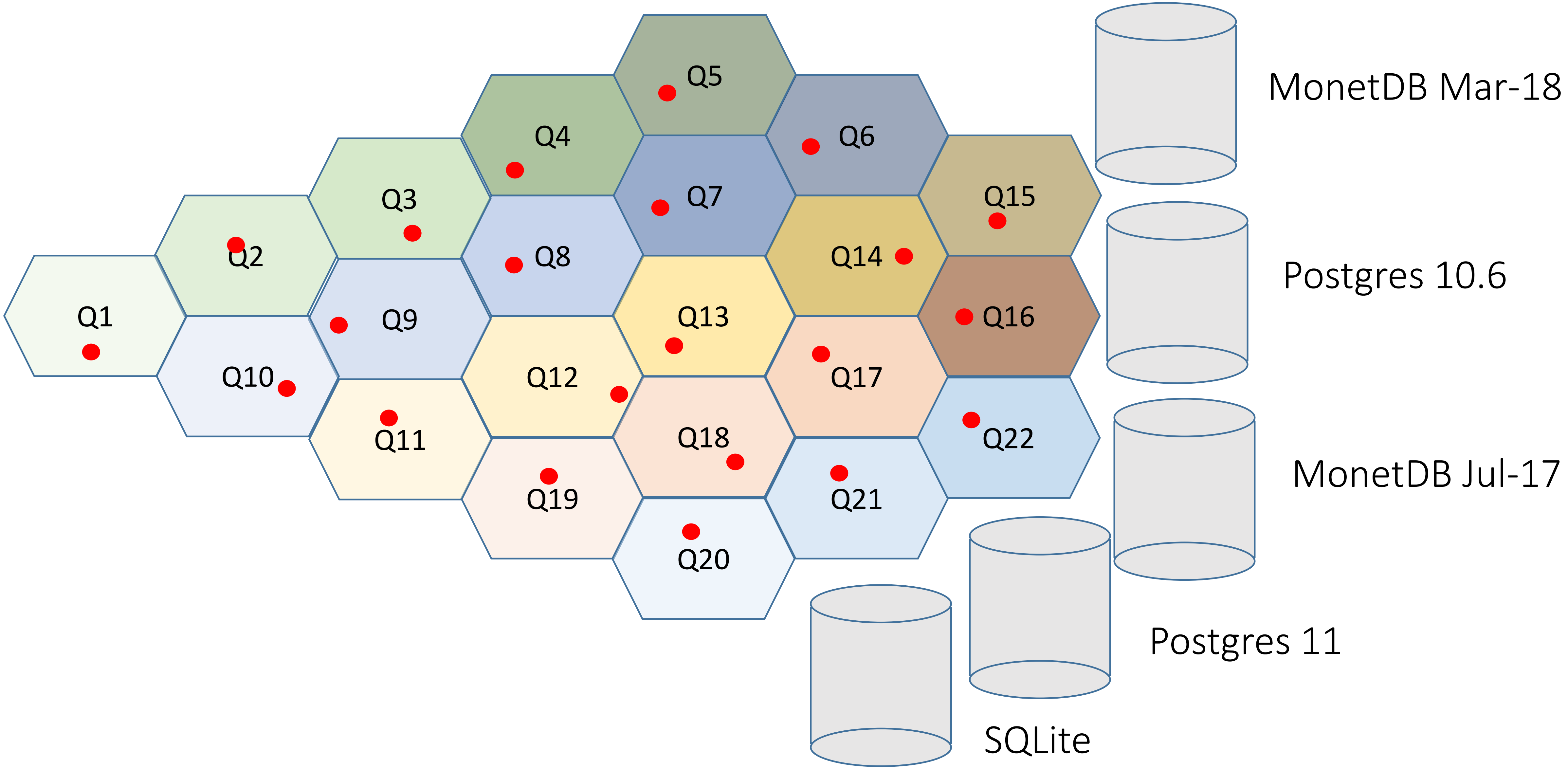
Postgres 10.6

MonetDB Jul-17

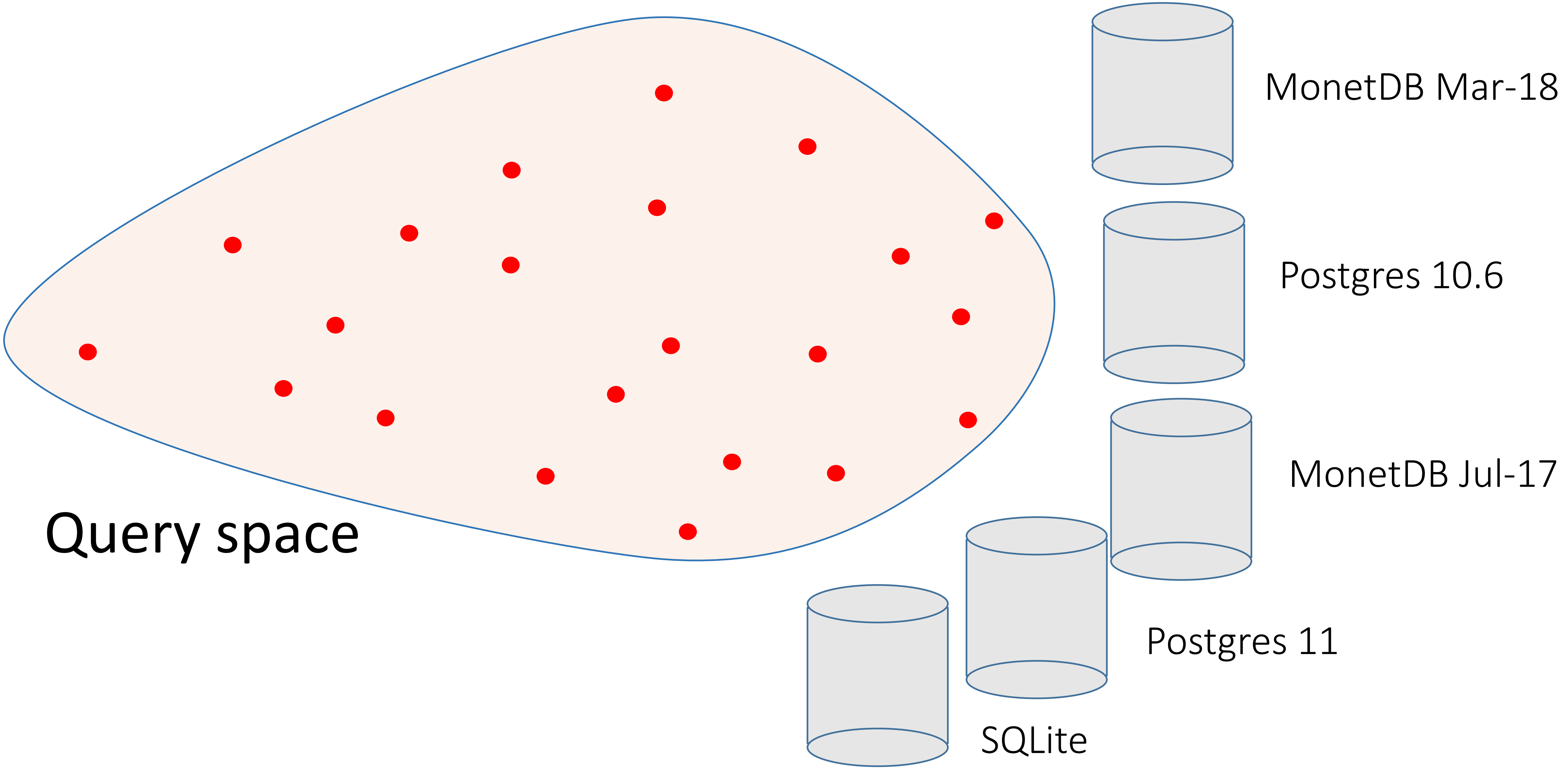
Postgres 11

SQLite

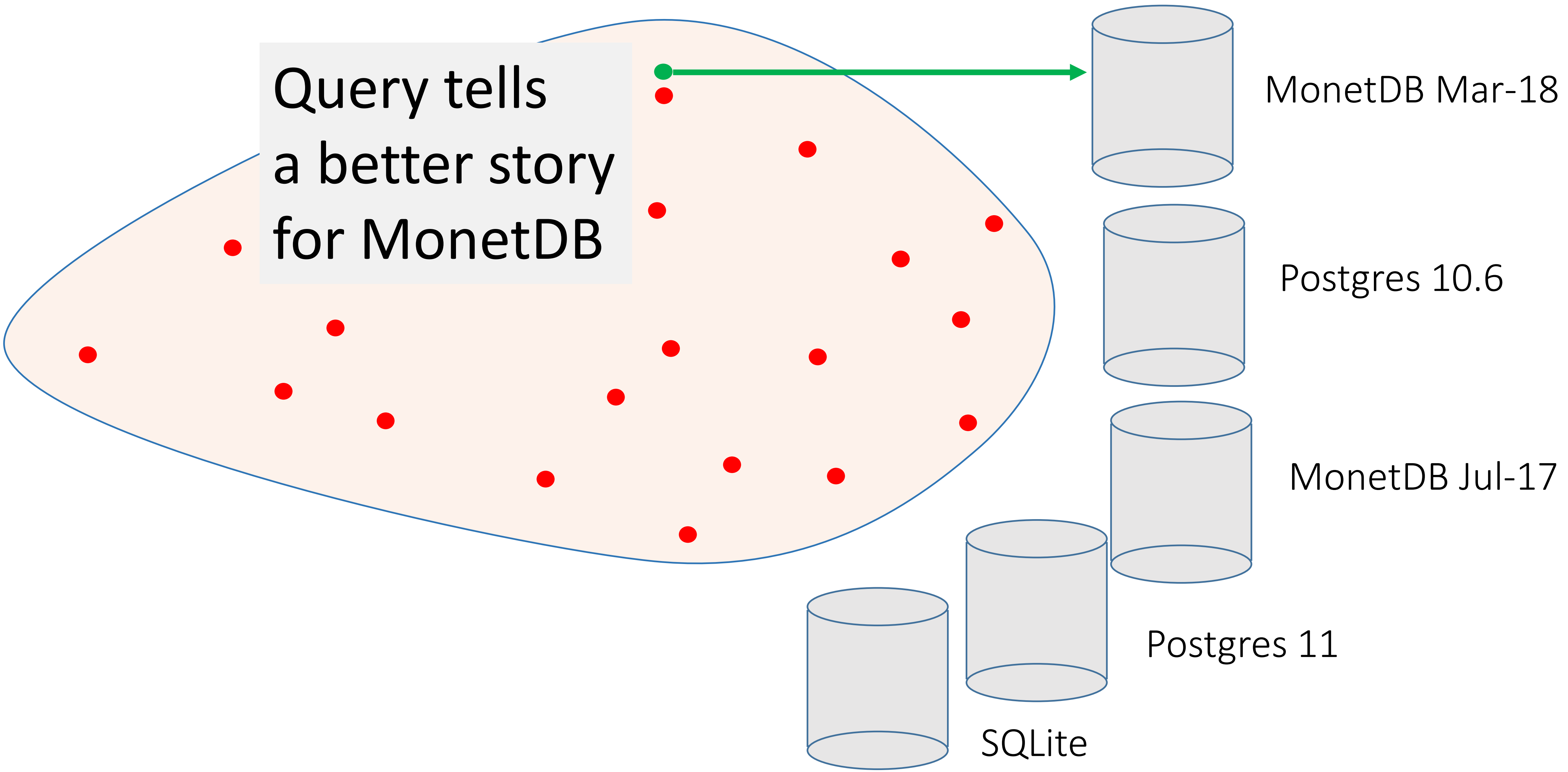
The Solution, TPC-H ?



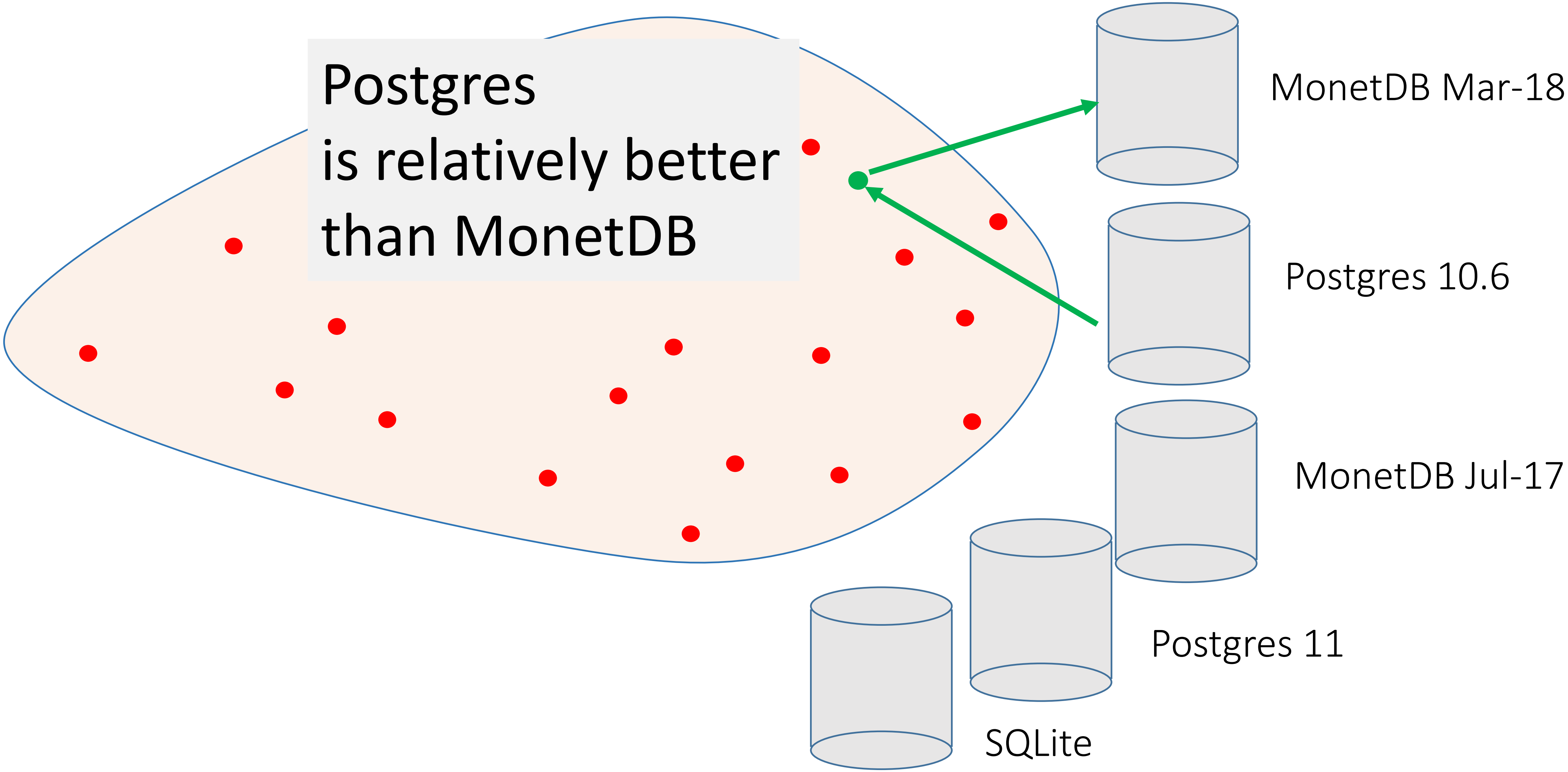
TPC-H is a collection of ‘random points’?



TPC-H may miss clarifying queries



TPC-H may miss discriminative queries



Postgres
is relatively better
than MonetDB

MonetDB Mar-18

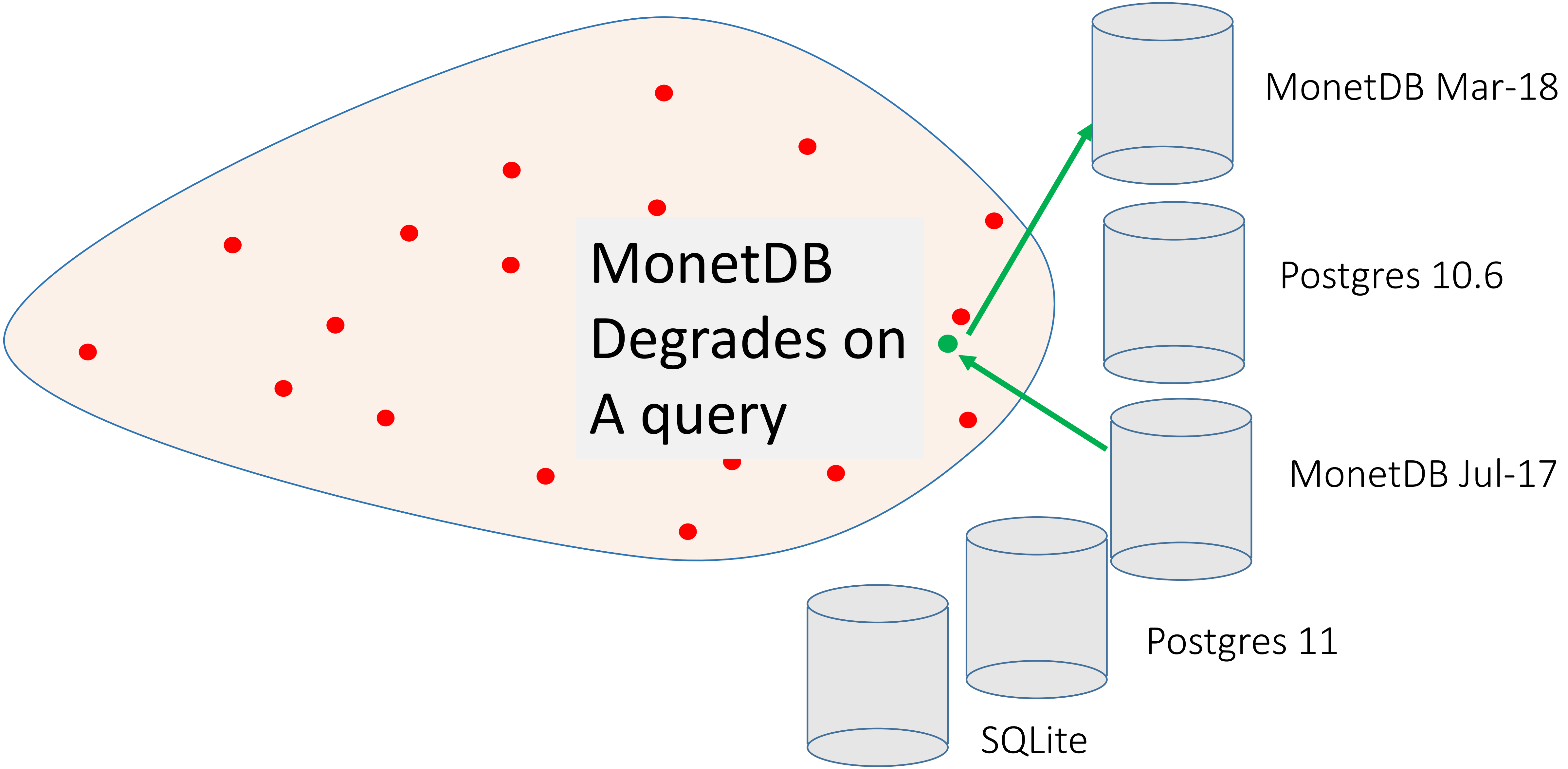
Postgres 10.6

MonetDB Jul-17

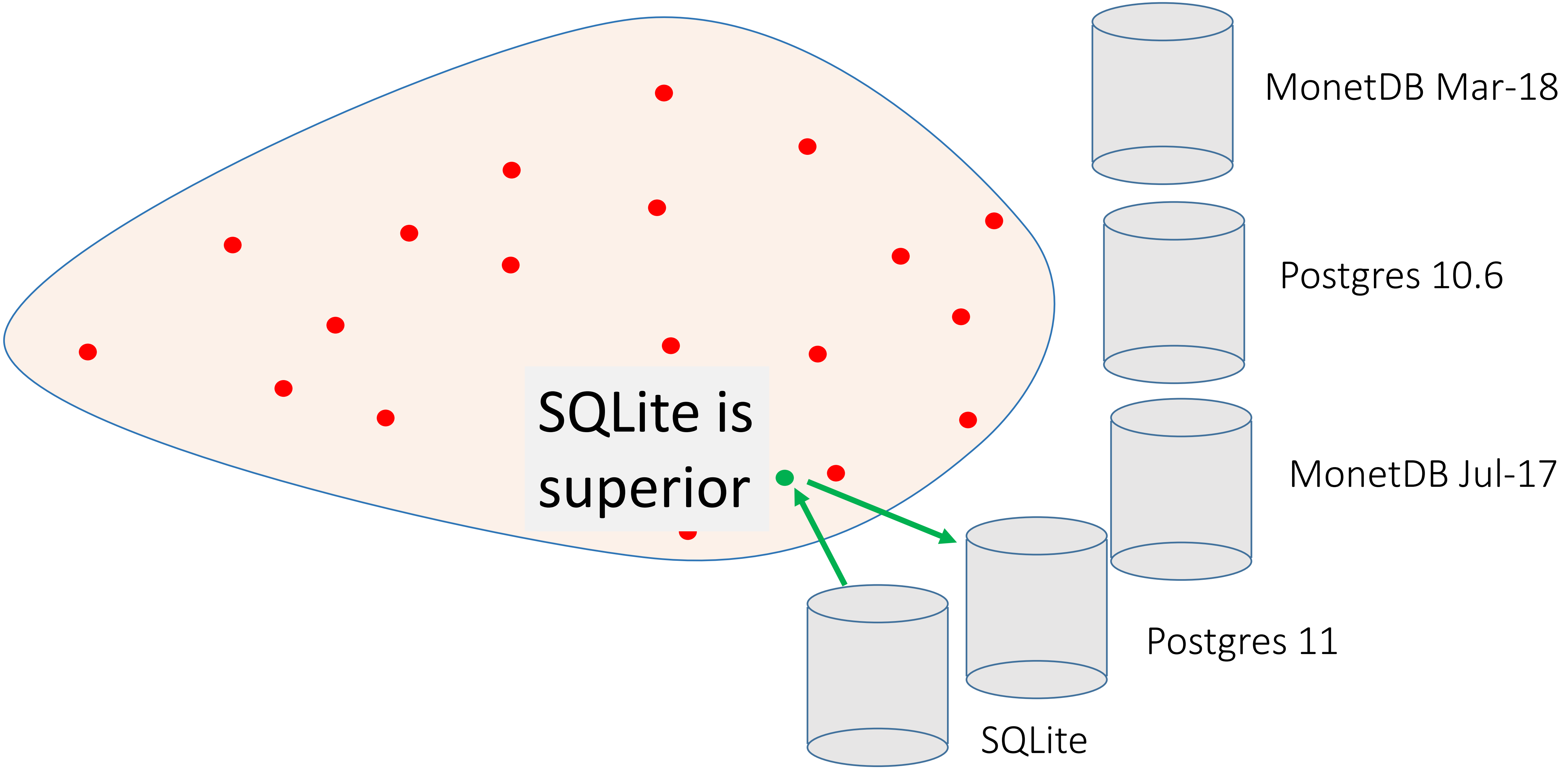
Postgres 11

SQLite

TPC-H may miss discriminative queries



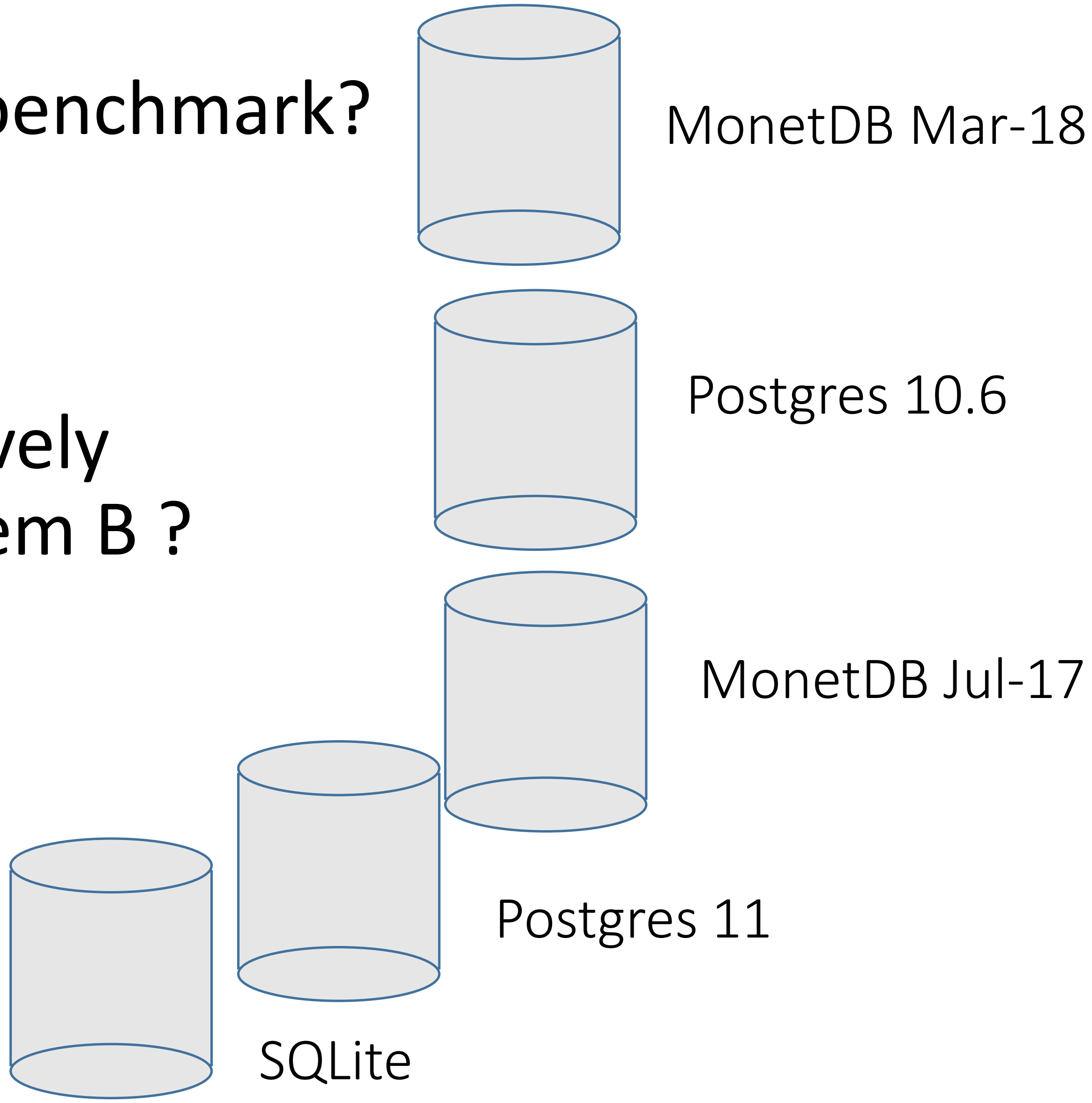
TPC-H may miss discriminative queries



The Challenge

- Which system is better on a benchmark?

- What queries perform relatively better on system A than system B ?



MonetDB Mar-18

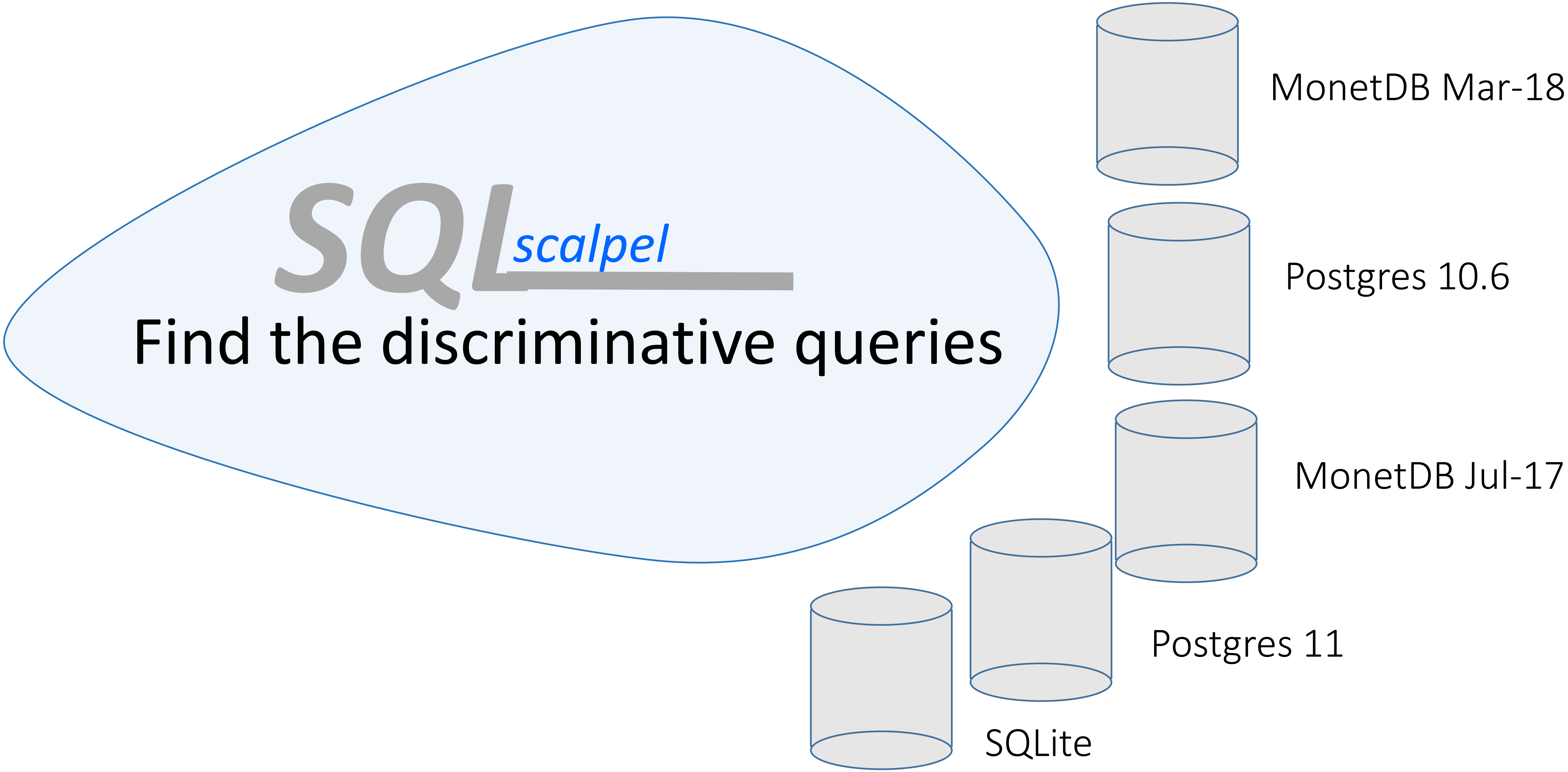
Postgres 10.6

MonetDB Jul-17

Postgres 11

SQLite

The Challenge



SQL*scalpel*

Find the discriminative queries

MonetDB Mar-18

Postgres 10.6

MonetDB Jul-17

Postgres 11

SQLite

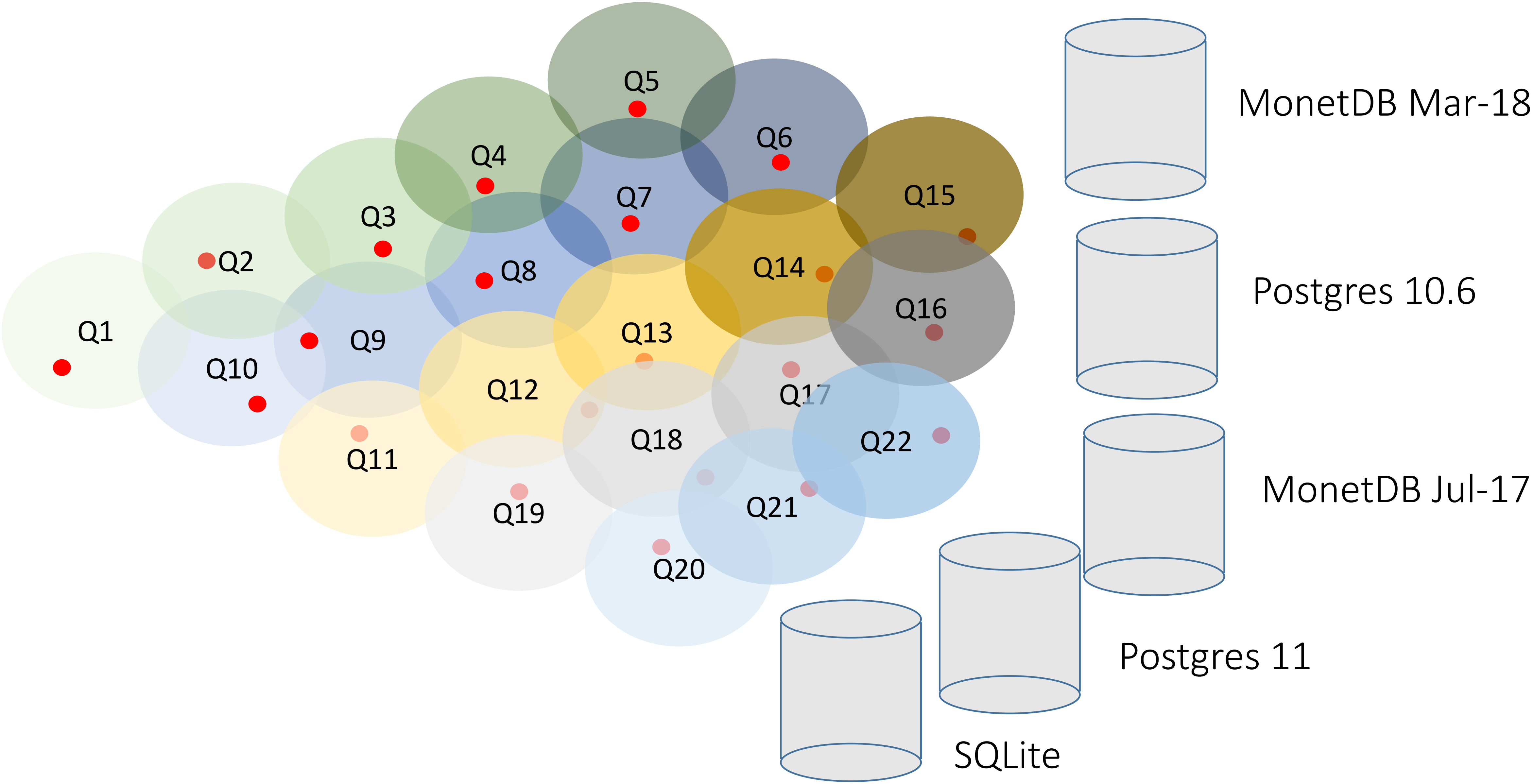
Assumptions

SQL*scalpel*

Find the discriminative queries

- The database schema and data distribution are given
- A collection of business inspired queries are available
- No direct access to the Database/DBMS/Platform

The Solution, TPC-H as a start



Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
Q9
Q10
Q11
Q12
Q13
Q14
Q15
Q16
Q17
Q18
Q19
Q20
Q21
Q22

MonetDB Mar-18

Postgres 10.6

MonetDB Jul-17

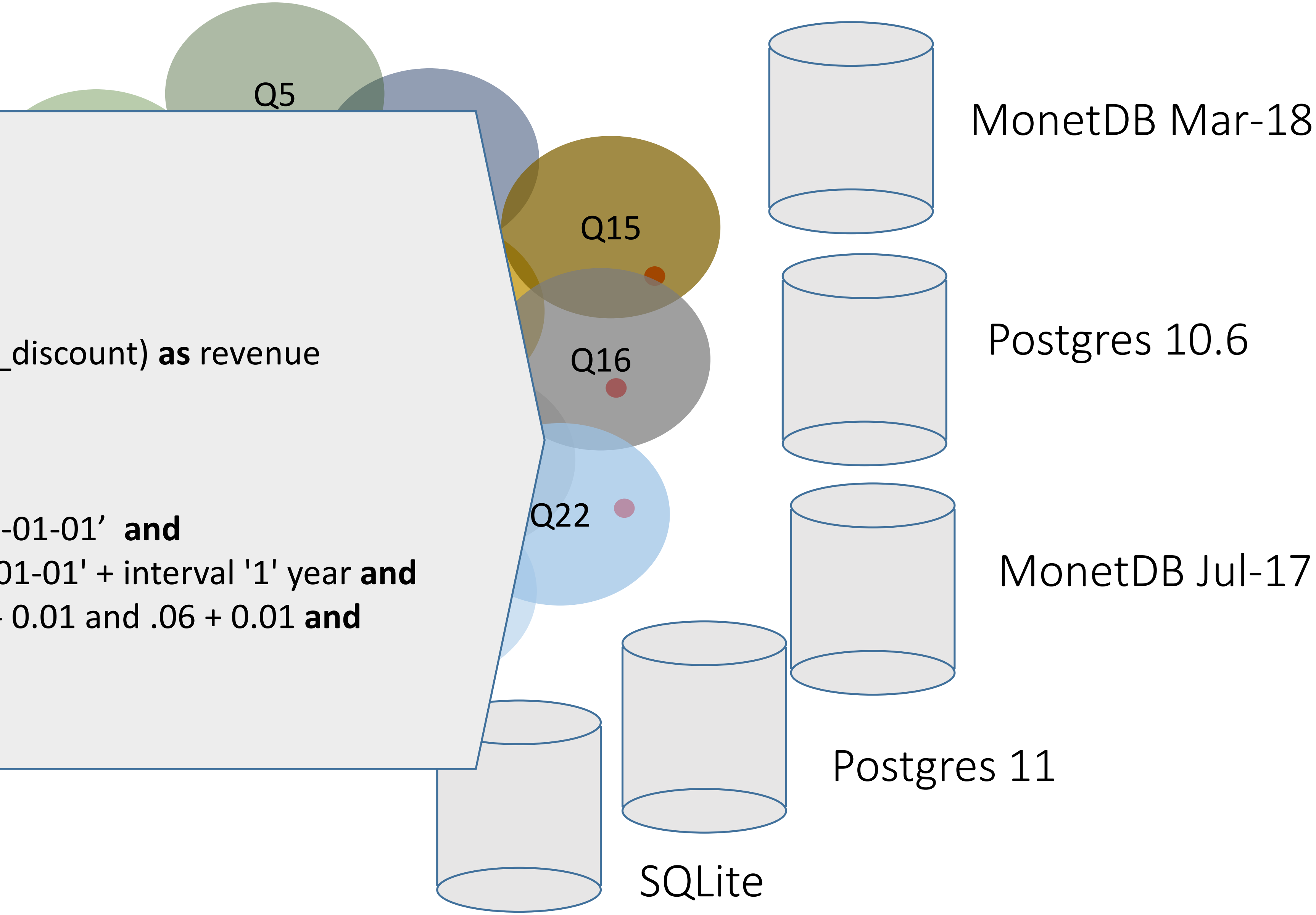
Postgres 11

SQLite

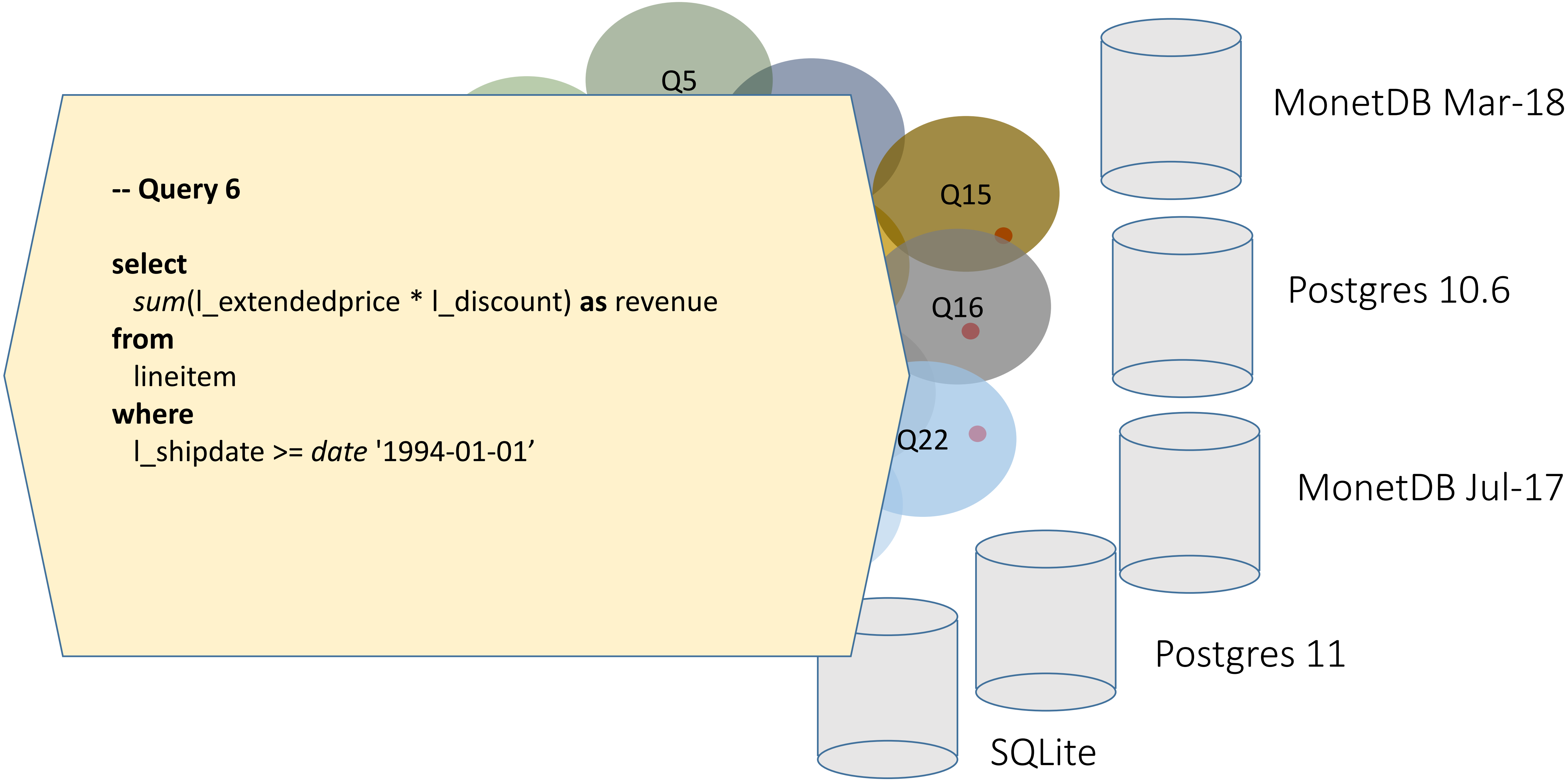
The Solution, TPC-H as a start

```
-- Query 6

select
  sum(l_extendedprice * l_discount) as revenue
from
  lineitem
where
  l_shipdate >= date '1994-01-01’  and
  l_shipdate < date '1994-01-01' + interval '1' year and
  l_discount between .06 - 0.01 and .06 + 0.01 and
  l_quantity < 24;
```



The Solution, TPC-H as a start



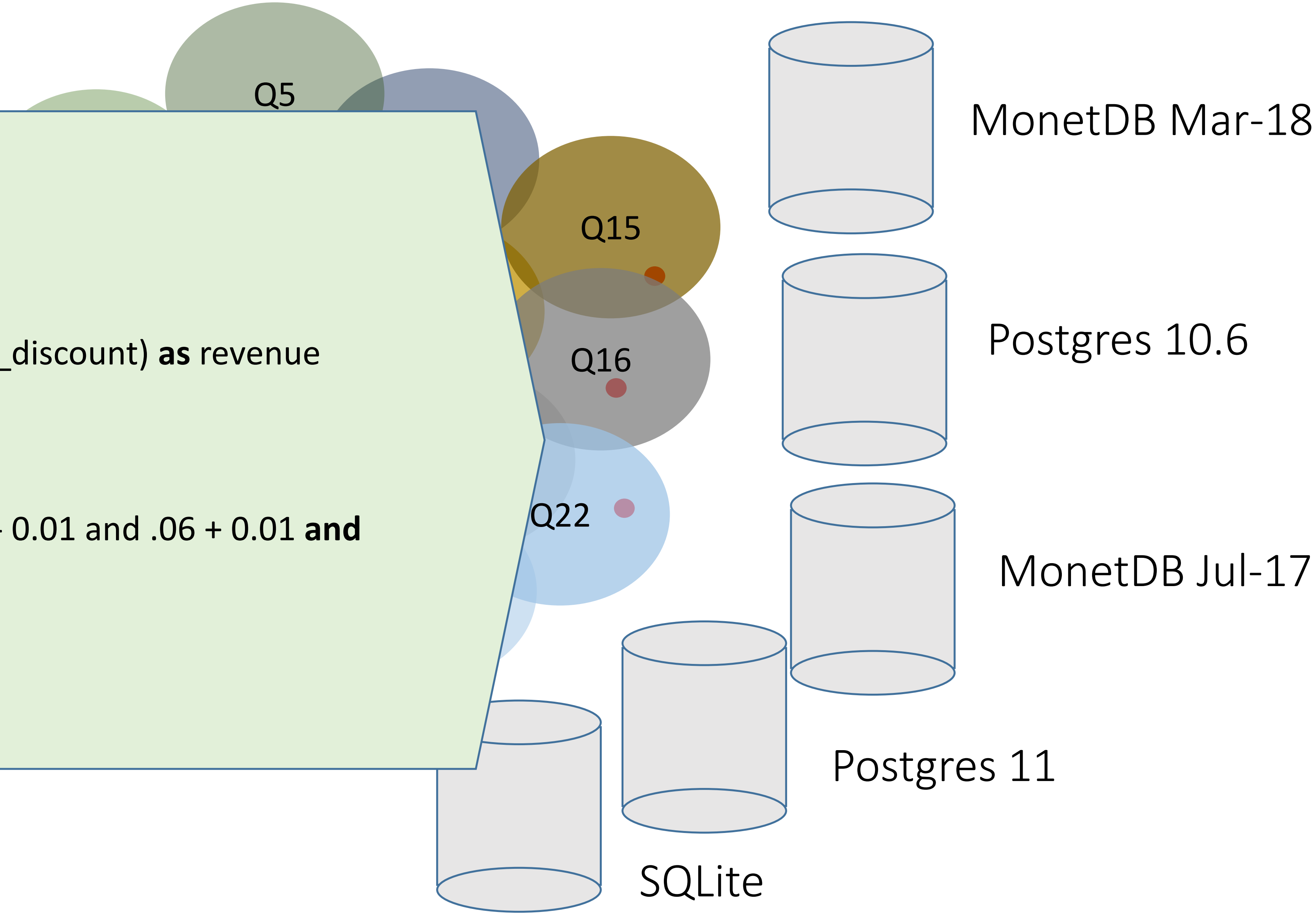
```
-- Query 6

select
  sum(l_extendedprice * l_discount) as revenue
from
  lineitem
where
  l_shipdate >= date '1994-01-01’
```

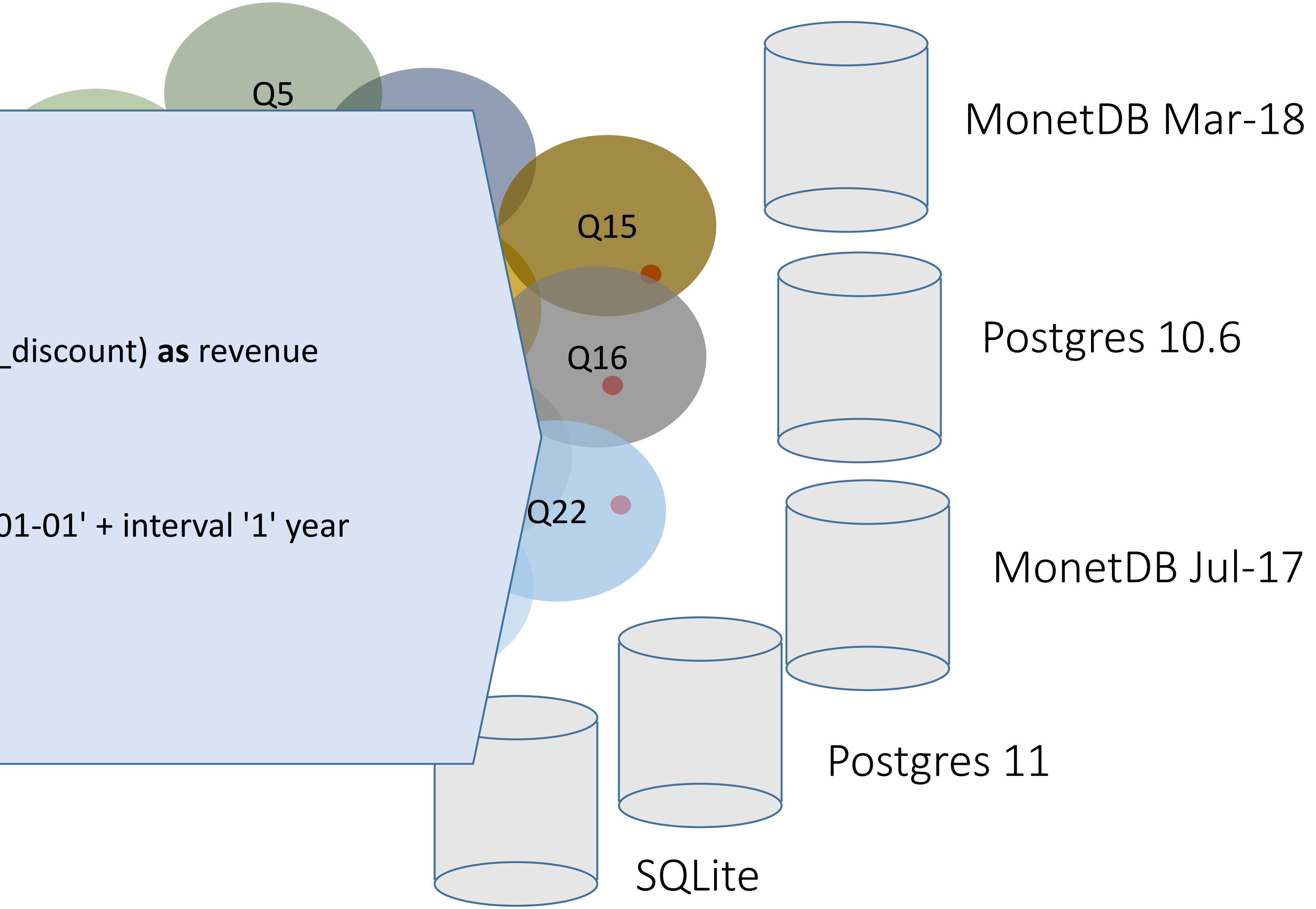
The Solution, TPC-H as a start

```
-- Query 6

select
  sum(l_extendedprice * l_discount) as revenue
from
  lineitem
where
  l_discount between .06 - 0.01 and .06 + 0.01 and
  l_quantity < 24;
```



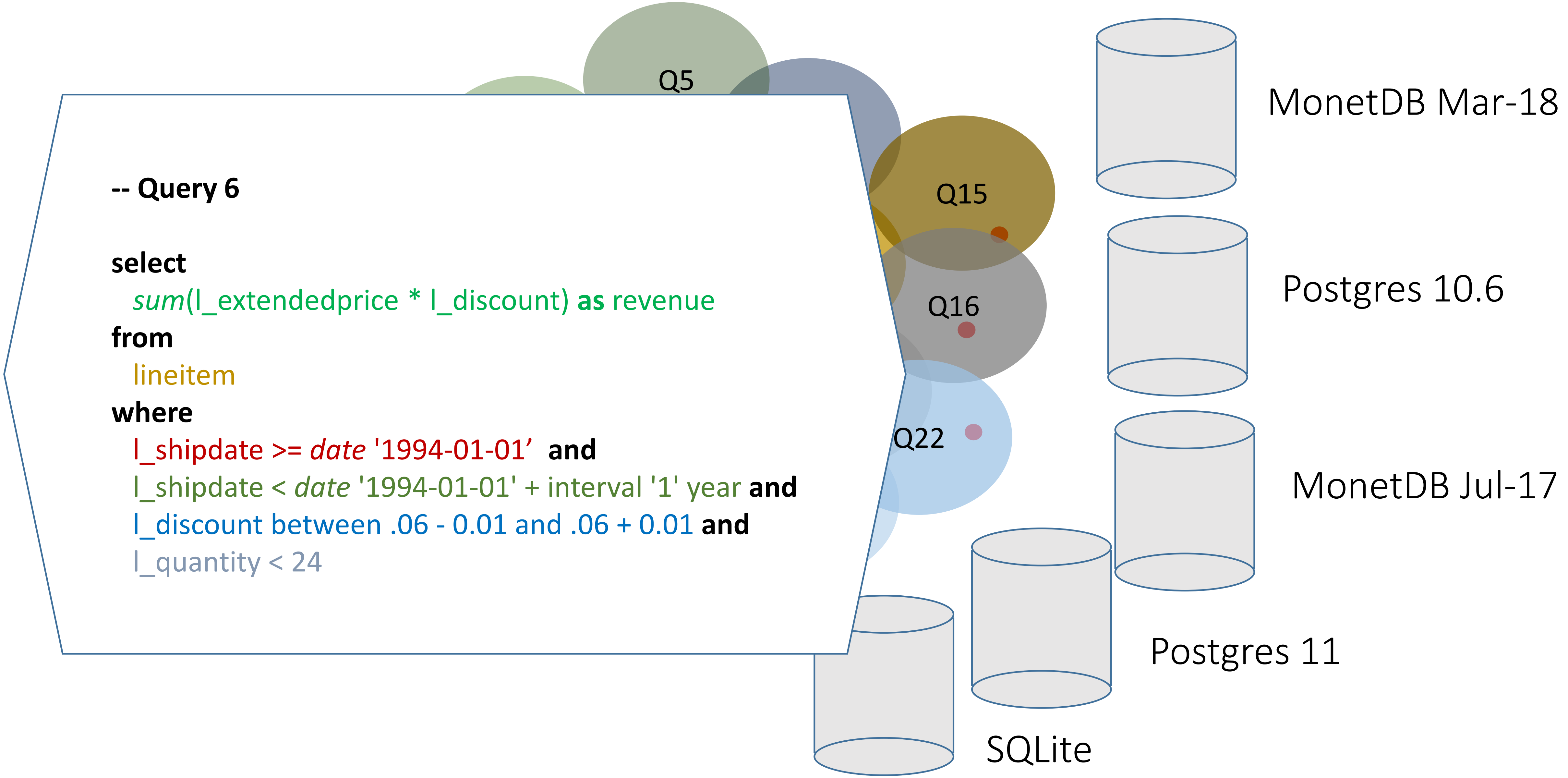
The Solution, TPC-H as a start



```
-- Query 6

select
  sum(l_extendedprice * l_discount) as revenue
from
  lineitem
where
  l_shipdate < date '1994-01-01' + interval '1' year
```

The Solution, TPC-H as a start



```
-- Query 6

select
  sum(l_extendedprice * l_discount) as revenue
from
  lineitem
where
  l_shipdate >= date '1994-01-01’  and
  l_shipdate < date '1994-01-01' + interval '1' year and
  l_discount between .06 - 0.01 and .06 + 0.01 and
  l_quantity < 24
```

monetdb solutions

SQLscalpel compiles the query into a grammar

```
-- Query 6

select
  sum(l_extendedprice * l_discount) as revenue
from
  lineitem
where
  l_shipdate >= date '1994-01-01’  and
  l_shipdate < date '1994-01-01' + interval '1' year and
  l_discount between .06 - 0.01 and .06 + 0.01 and
  l_quantity < 24
```

```
tpch_q6:
  SELECT ${projection} FROM ${table}
  WHERE ${pred} ${predlist}*

projection:
  sum(l_extendedprice * l_discount) as revenue

table:
 lineitem

pred:
 l_shipdate >= date '1994-01-01'
 l_shipdate < date '1994-01-01' + interval '1' year
 l_discount between 0.06 - 0.01 and 0.06 + 0.01
 l_quantity < 24

predlist:
    AND ${pred}
```

SQLscalpel enumerates templates

```
SELECT ${projection} FROM ${table}
  WHERE ${pred}

SELECT ${projection} FROM ${table}
  WHERE ${pred} AND ${pred}

SELECT ${projection} FROM ${table}
  WHERE ${pred} AND ${pred} AND ${pred}


SELECT ${projection} FROM ${table}
  WHERE ${pred} AND ${pred} AND ${pred} AND ${pred}
```

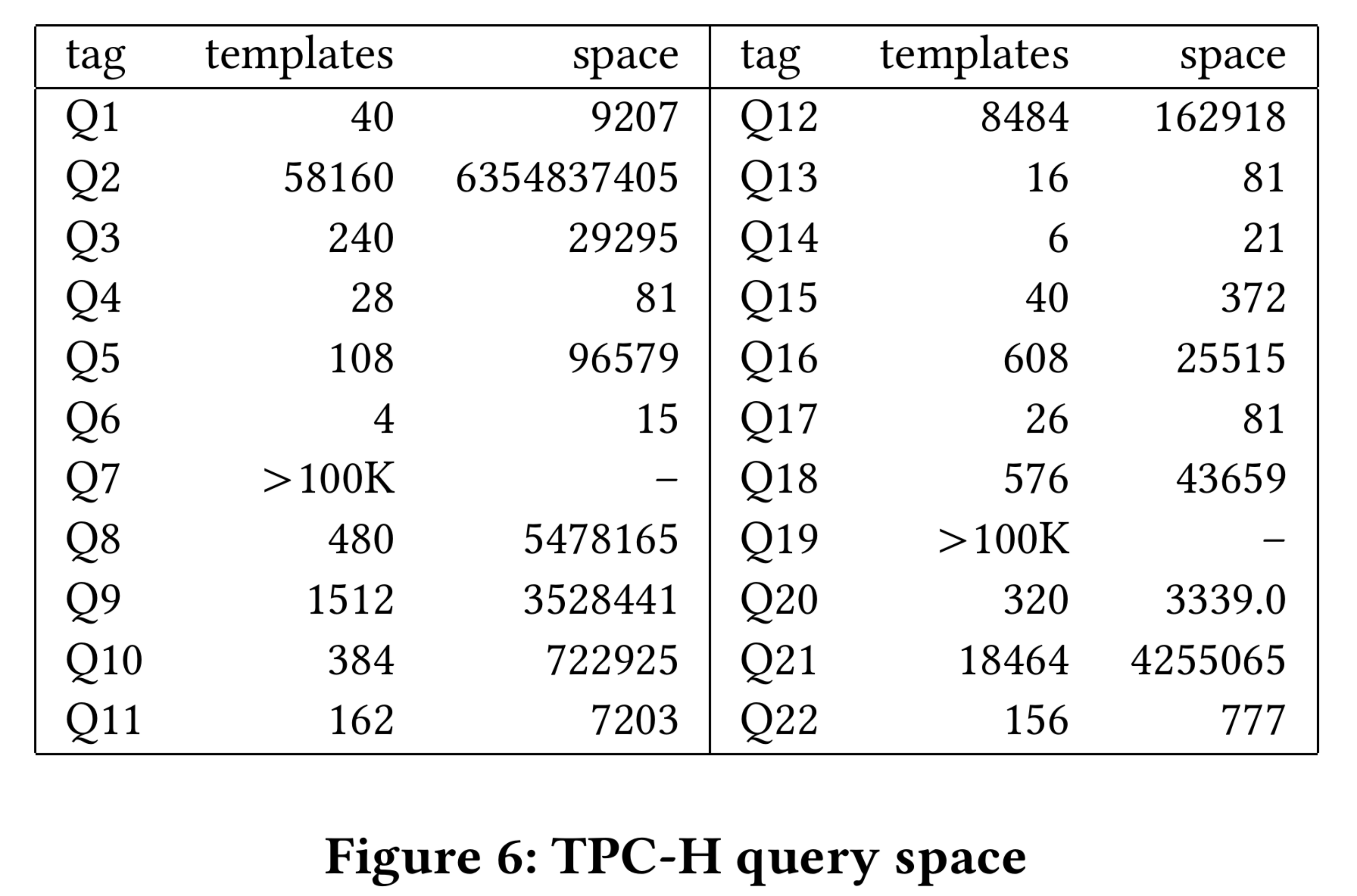
```
tpch_q6:
  SELECT ${projection} FROM ${table}
  WHERE ${pred} ${predlist}*

projection:
  sum(l_extendedprice * l_discount) as revenue

table:
 lineitem

pred:
 l_shipdate >= date '1994-01-01'
 l_shipdate < date '1994-01-01' + interval '1' year
 l_discount between 0.06 - 0.01 and 0.06 + 0.01
 l_quantity < 24

predlist:
   AND ${pred}
```



tag	templates	space	tag	templates	space
Q1	40	9207	Q12	8484	162918
Q2	58160	6354837405	Q13	16	81
Q3	240	29295	Q14	6	21
Q4	28	81	Q15	40	372
Q5	108	96579	Q16	608	25515
Q6	4	15	Q17	26	81
Q7	>100K	–	Q18	576	43659
Q8	480	5478165	Q19	>100K	–
Q9	1512	3528441	Q20	320	3339.0
Q10	384	722925	Q21	18464	4255065
Q11	162	7203	Q22	156	777

Figure 6: TPC-H query space

Architecture

Maintain a query pool

Pick promising candidates

Keep a workflow database

Support analysis

SQL scalpel

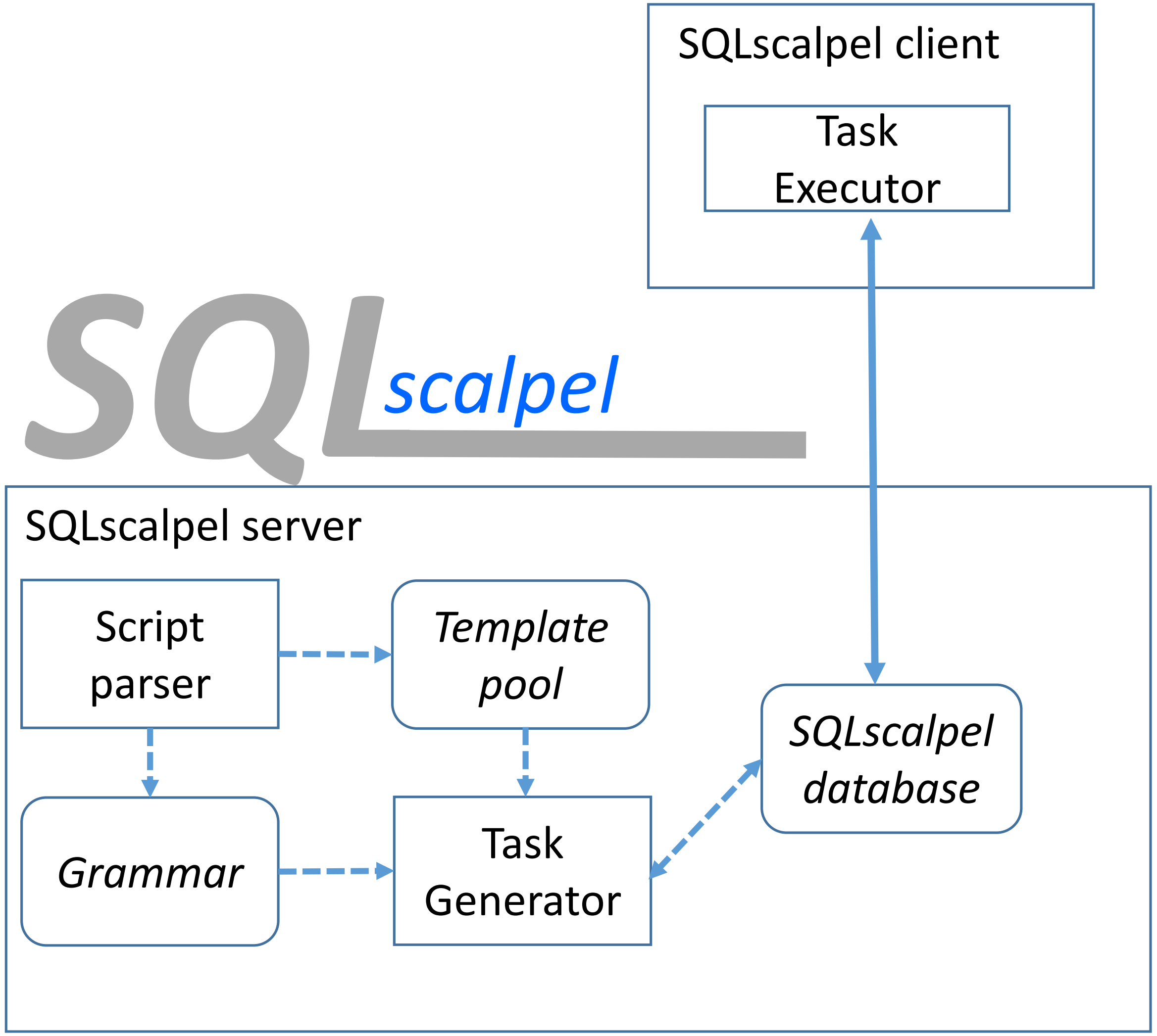
SQLscalpel architecture

Maintain a query pool

Pick promising candidates

Keep a workflow database

Support analysis



SQLscalpel architecture

Maintain a query pool → Strategies:

- Baseline
- Random
- Alter a lexical term
- Expand a query
- Prune a query
- Ditch manually

Pick promising candidates

Keep a workflow database

Support analysis

SQLscalpel architecture

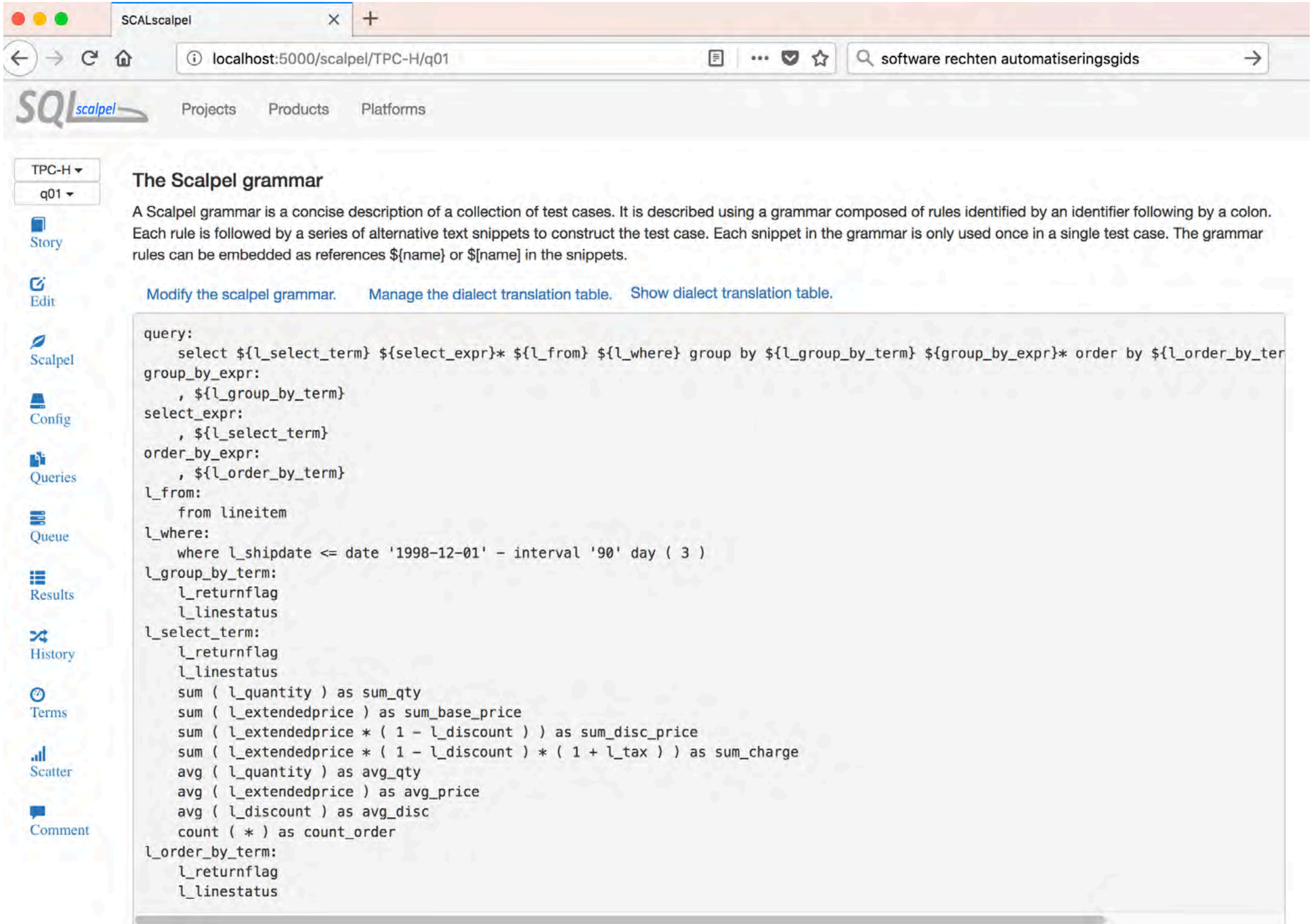
Maintain a query pool

Pick promising candidates → Strategies:

- FCFS
- Manual steering
- Simulated annealing
- Biased terms

Keep a workflow database

Support analysis

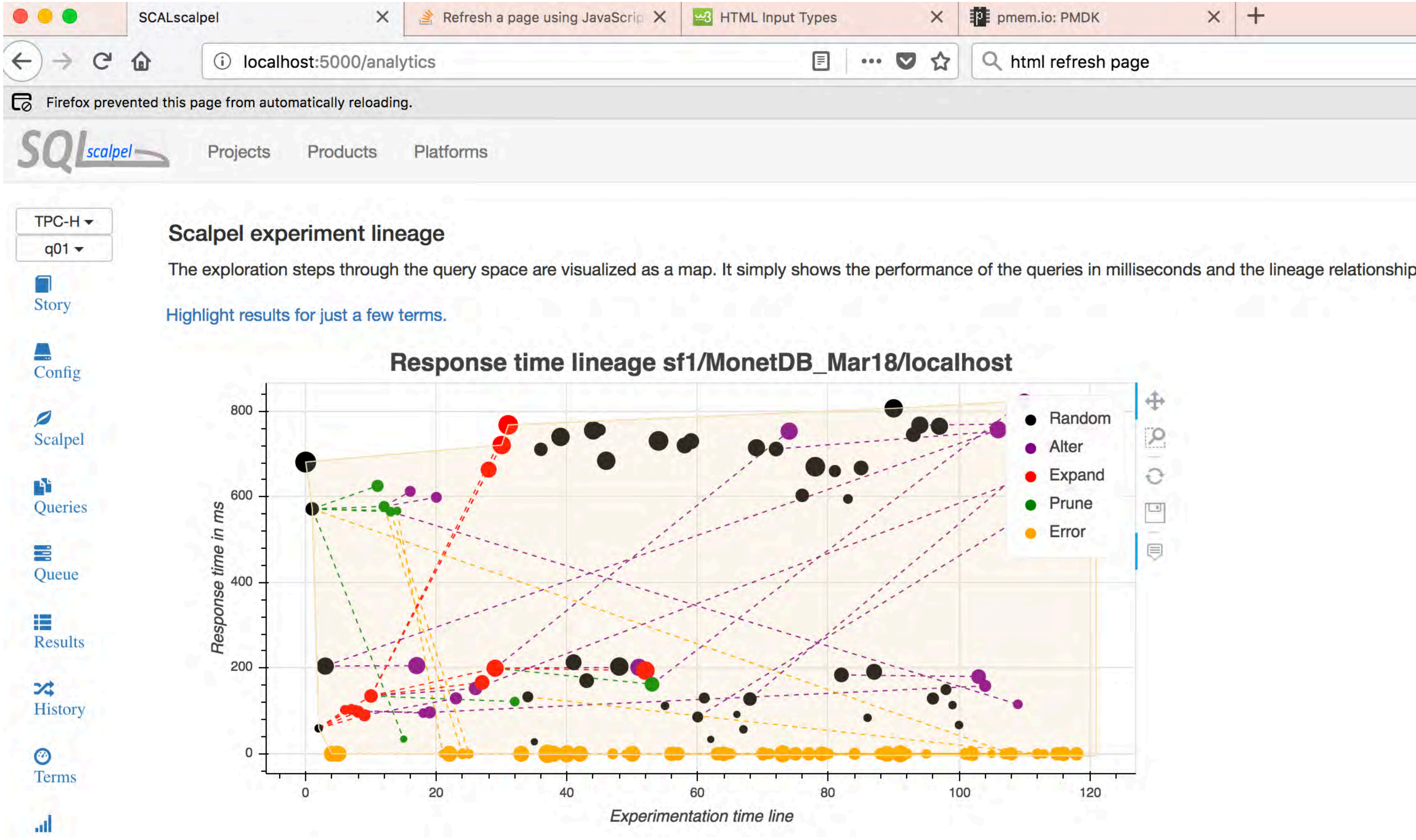


The Scalpel grammar

A Scalpel grammar is a concise description of a collection of test cases. It is described using a grammar composed of rules identified by an identifier following by a colon. Each rule is followed by a series of alternative text snippets to construct the test case. Each snippet in the grammar is only used once in a single test case. The grammar rules can be embedded as references ${name} or $[name] in the snippets.

Modify the scalpel grammar.   Manage the dialect translation table.   Show dialect translation table.

```
query:
    select ${l_select_term} ${select_expr}* ${l_from} ${l_where} group by ${l_group_by_term} ${group_by_expr}* order by ${l_order_by_ter
group_by_expr:
    , ${l_group_by_term}
select_expr:
    , ${l_select_term}
order_by_expr:
    , ${l_order_by_term}
l_from:
    from lineitem
l_where:
    where l_shipdate <= date '1998-12-01' - interval '90' day ( 3 )
l_group_by_term:
    l_returnflag
    l_linestatus
l_select_term:
    l_returnflag
    l_linestatus
    sum ( l_quantity ) as sum_qty
    sum ( l_extendedprice ) as sum_base_price
    sum ( l_extendedprice * ( 1 - l_discount ) ) as sum_disc_price
    sum ( l_extendedprice * ( 1 - l_discount ) * ( 1 + l_tax ) ) as sum_charge
    avg ( l_quantity ) as avg_qty
    avg ( l_extendedprice ) as avg_price
    avg ( l_discount ) as avg_disc
    count ( * ) as count_order
l_order_by_term:
    l_returnflag
    l_linestatus
```



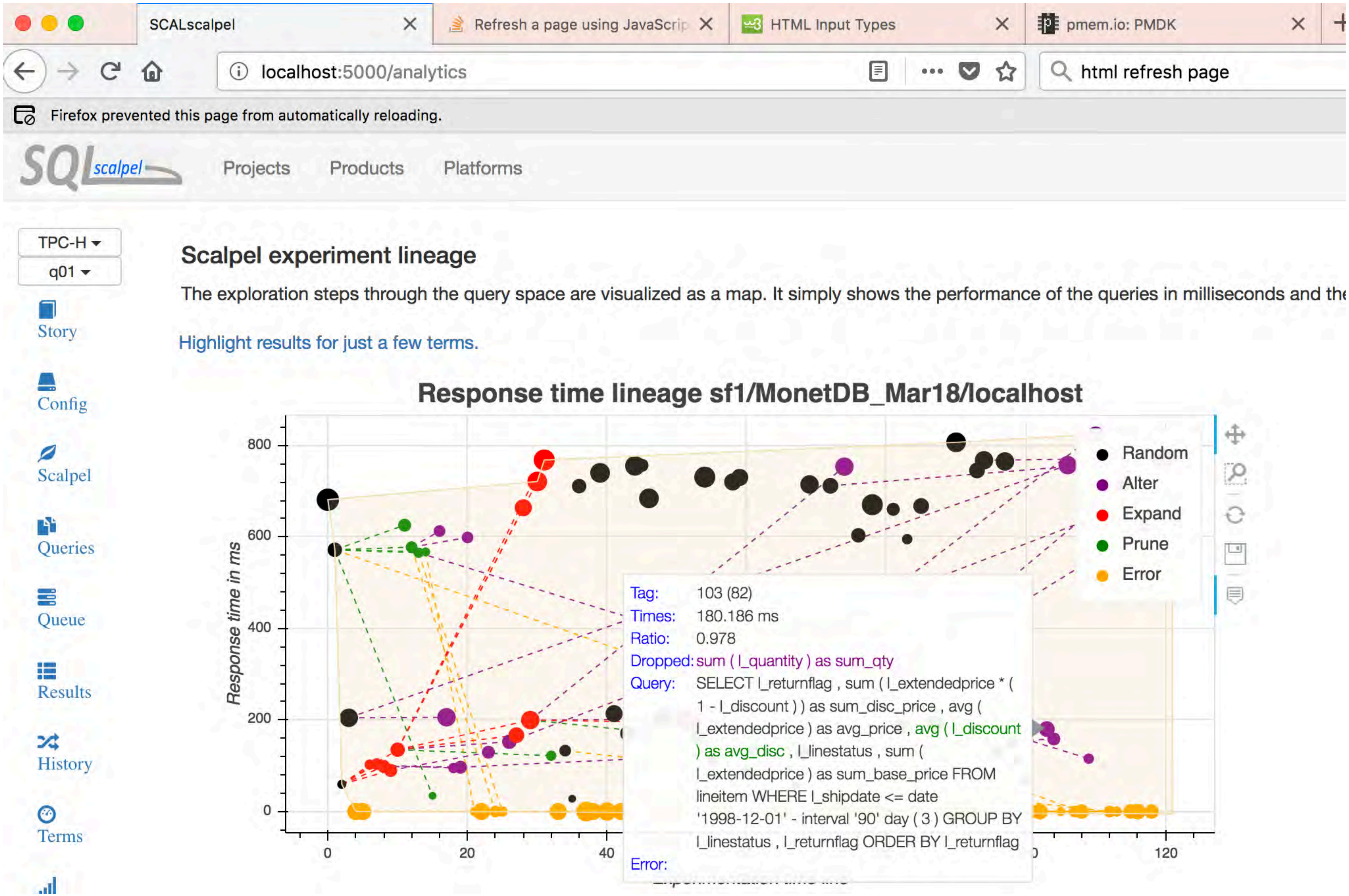
Firefox prevented this page from automatically reloading.

Scalpel experiment lineage

The exploration steps through the query space are visualized as a map. It simply shows the performance of the queries in milliseconds and the lineage relationship

Highlight results for just a few terms.

Response time lineage sf1/MonetDB_Mar18/localhost monetdb solutions



Firefox prevented this page from automatically reloading.

SQLscalpel

Scalpel experiment lineage

The exploration steps through the query space are visualized as a map. It simply shows the performance of the queries in milliseconds and the

Highlight results for just a few terms.

Response time lineage sf1/MonetDB_Mar18/localhost

Tag: 103 (82)

Times: 180.186 ms

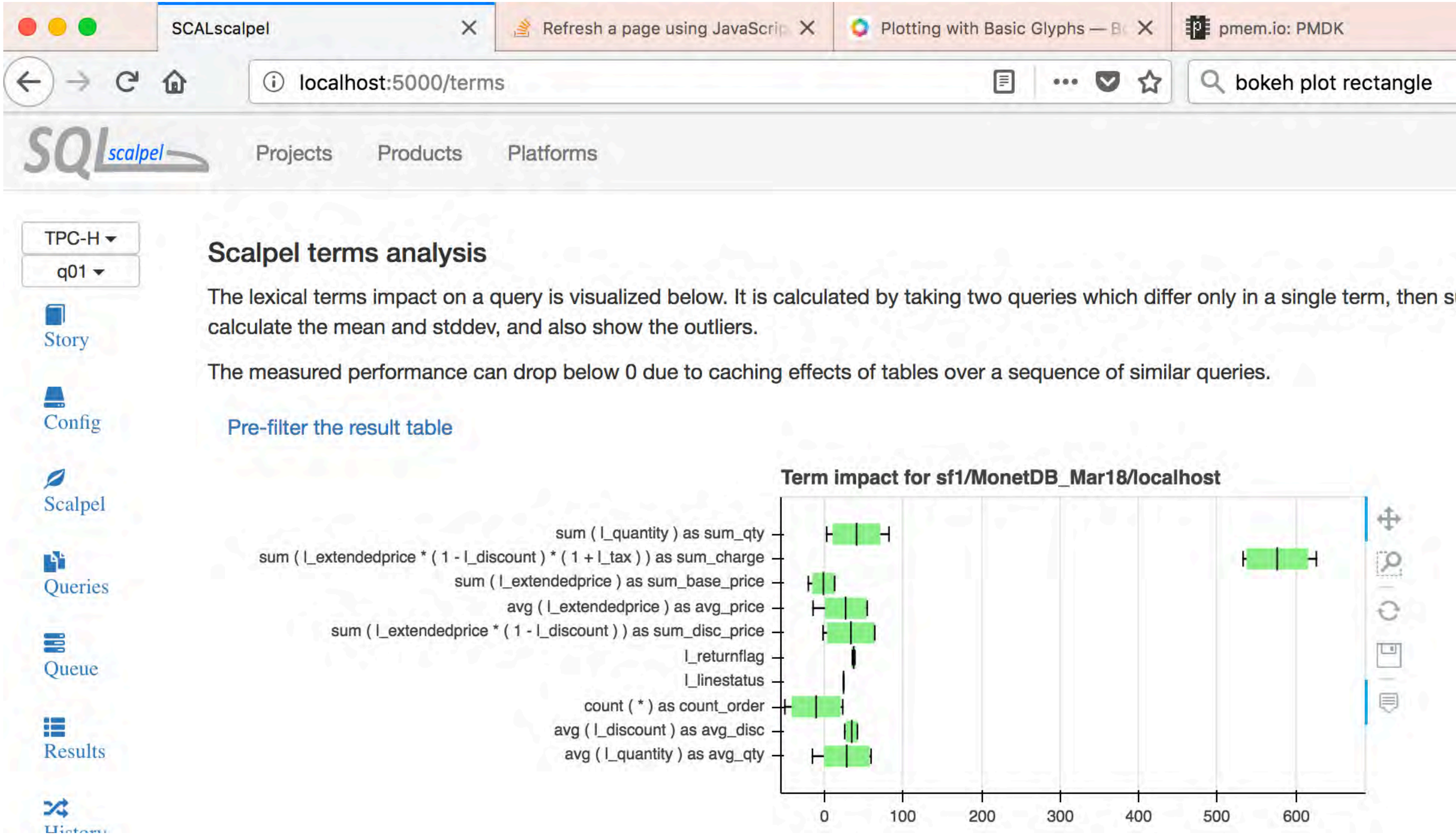
Ratio: 0.978

Dropped: sum (l_quantity) as sum_qty

Query: SELECT l_returnflag , sum (l_extendedprice * (1 - l_discount)) as sum_disc_price , avg (l_extendedprice) as avg_price , avg (l_discount) as avg_disc , l_linestatus , sum (l_extendedprice) as sum_base_price FROM lineitem WHERE l_shipdate <= date '1998-12-01' - interval '90' day (3) GROUP BY l_linestatus , l_returnflag ORDER BY l_returnflag

Error:

monetdb
solutions



Scalpel terms analysis

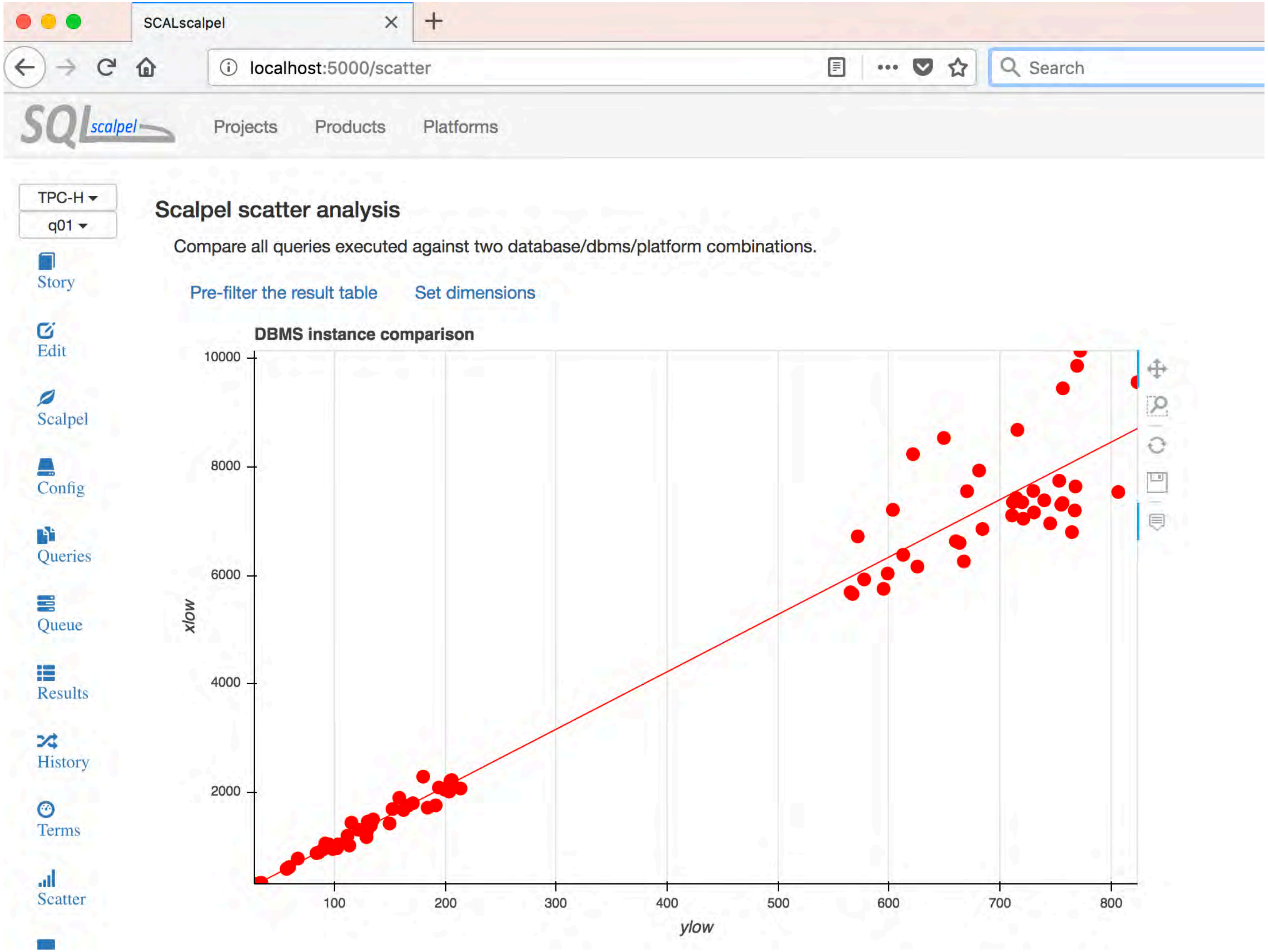
The lexical terms impact on a query is visualized below. It is calculated by taking two queries which differ only in a single term, then s
calculate the mean and stddev, and also show the outliers.

The measured performance can drop below 0 due to caching effects of tables over a sequence of similar queries.

Pre-filter the result table

Term impact for sf1/MonetDB_Mar18/localhost

sum (l_quantity) as sum_qty
sum (l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge
sum (l_extendedprice) as sum_base_price
avg (l_extendedprice) as avg_price
sum (l_extendedprice * (1 - l_discount)) as sum_disc_price
l_returnflag
l_linestatus
count (*) as count_order
avg (l_discount) as avg_disc
avg (l_quantity) as avg_qty
0
100
200
300
400
500
600



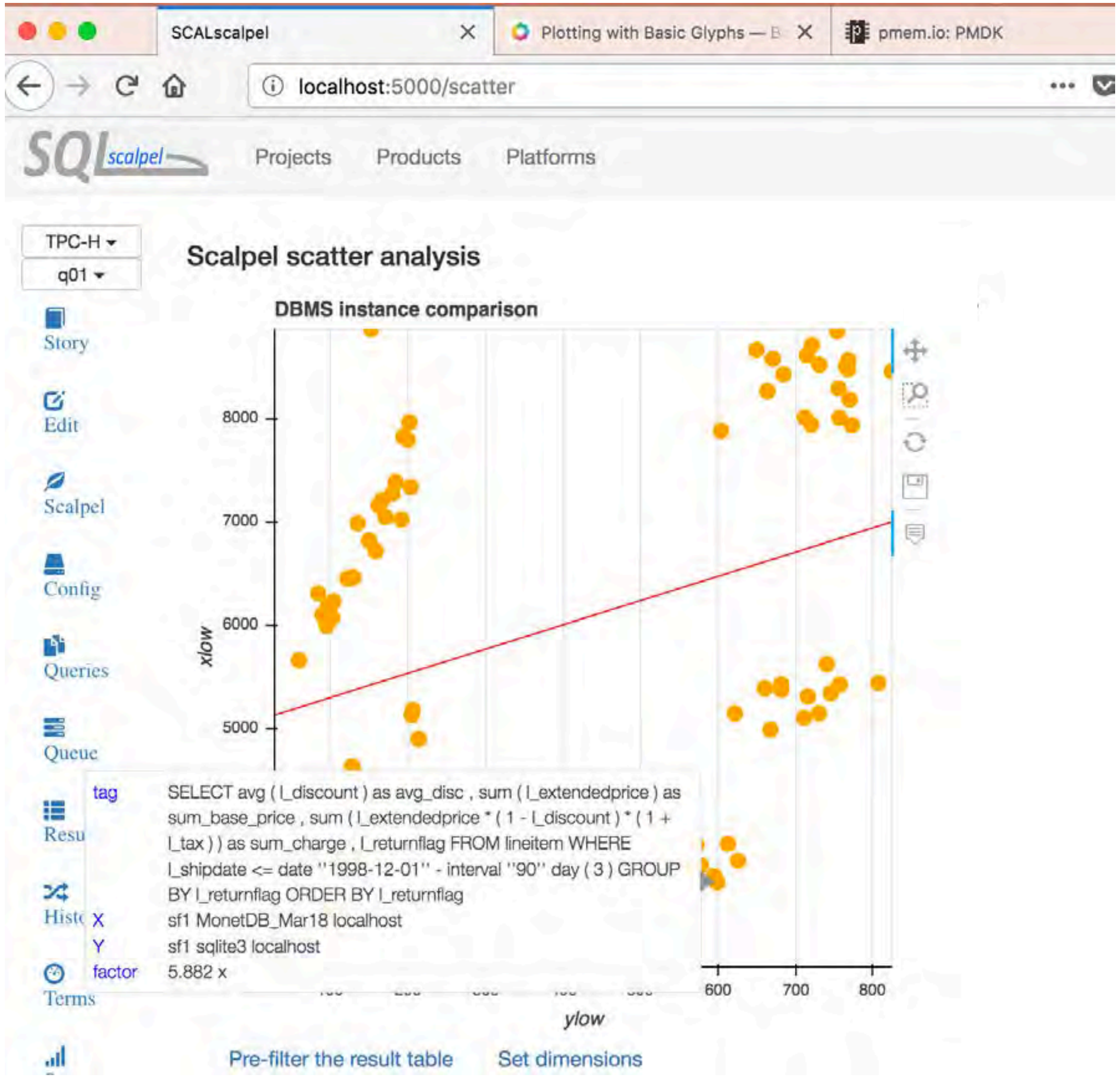
Scalpel scatter analysis

Compare all queries executed against two database/dbms/platform combinations.

Pre-filter the result table

Set dimensions

DBMS instance comparison



Scalpel scatter analysis

tag	SELECT avg (l_discount) as avg_disc , sum (l_extendedprice) as sum_base_price , sum (l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge , l_returnflag FROM lineitem WHERE l_shipdate <= date ''1998-12-01'' - interval ''90'' day (3) GROUP BY l_returnflag ORDER BY l_returnflag
X	sf1 MonetDB_Mar18 localhost
Y	sf1 sqlite3 localhost
factor	5.882 x

SQLscalpel pitfalls

Handle SQL dialects

Easily generate erroneous queries

Brings the target down using Cartesian results

```
tpch_q6:
  SELECT ${projection} FROM ${table} WHERE ${pred} ${predlist}*

projection:
  sum(l_extendedprice * l_discount) as revenue

table:
 lineitem
 lineitem, regions

pred:
 l_shipdate >= date '1994-01-01'
 l_shipdate < date '1994-01-01' + interval '1' year
 l_discount between 0.06 - 0.01 and 0.06 + 0.01
 l_quantity < 24

predlist:
    AND ${pred}
```

- SQLscalpel prototype is up and runr
  - Full-stack infrastructure
  - Drivers for MonetDB, Lite, Postgres
- Functional enhar d:
  - Multi-pa te projects
  - Built-in foru re the story

Questions

Divergence

Q1: SELECT count(*) FROM nation WHERE nation.n_name='BRAZIL'

Q2: SELECT count(*) FROM nation WHERE nation.n_name='BRAZIL’ AND nation.n_regionkey=1

$$\frac{\dfrac{T_A(Q_2)}{T_B(Q_2)}}{\dfrac{T_A(Q_1)}{T_B(Q_1)}} \Rightarrow \frac{T_A(Q_2)\;T_B(Q_1)}{T_B(Q_2)\;T_A(Q_1)}$$

Divergence

```
Q1: SELECT    count(*)  FROM nation  WHERE nation.n_name='BRAZIL'
Q2: SELECT    count(*)  FROM nation  WHERE nation.n_name='BRAZIL’ AND nation.n_regionkey=1
```

$$\frac{T_A(Q_2)\quad T_B(Q_1)}{T_B(Q_2)\quad T_A(Q_1)} \begin{cases} < 1 & \text{System A is better than B on } Q_2 \text{ against Q1} \\ = 1 & \\ > 1 & \text{System B is better than A on } Q_2 \text{ against Q1} \end{cases}$$