# Touchstone: Generating Enormous Query-Aware Test Databases

Yuming Li[1], Rong Zhang[1], Xiaoyan Yang[2],
Zhenjie Zhang[2], Aoying Zhou[1]

*[1]DaSE at East China Normal University*

*[2]Singapore R&D, Yitu Technology Ltd.*

# Test Databases Are Important!

- Application scenarios: DBMS testing, database application testing, application-driven benchmarking.

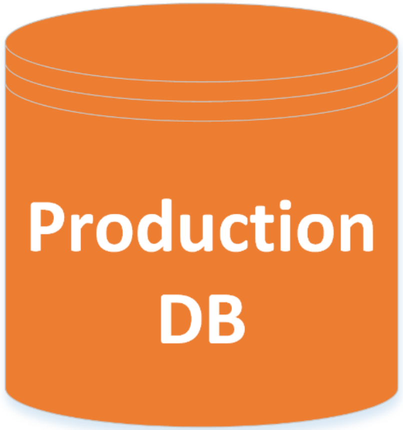
# Test Databases Are Important!

IT company
Solution provider

- Application scenarios: DBMS testing, database application testing, application-driven benchmarking.

# Test Databases Are Important!

IT company
Solution provider

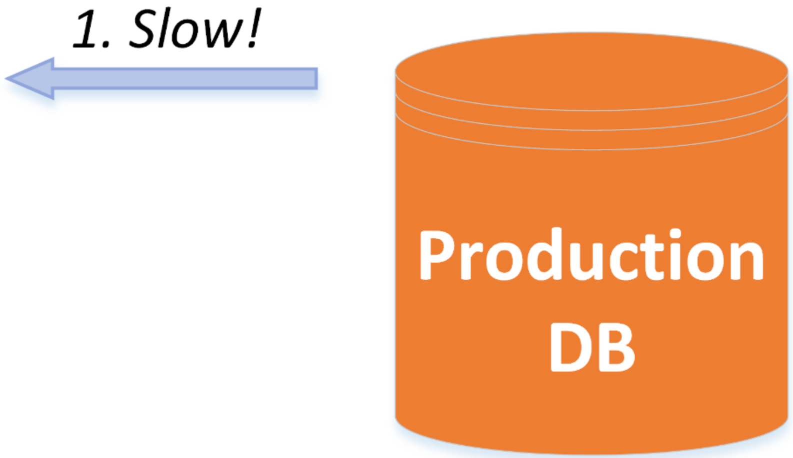



Bank
Application

- Application scenarios: DBMS testing, database application testing, application-driven benchmarking.

# Test Databases Are Important!

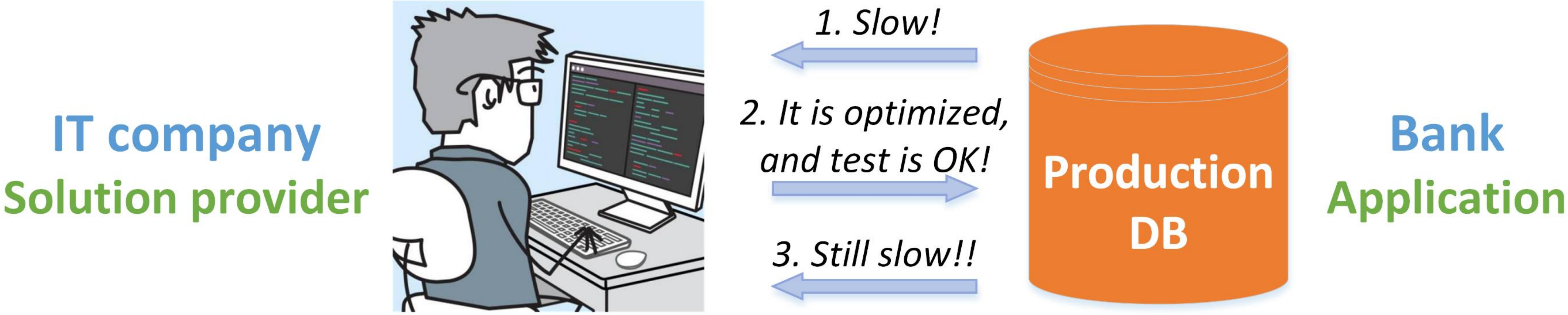
IT company
Solution provider




Bank
Application

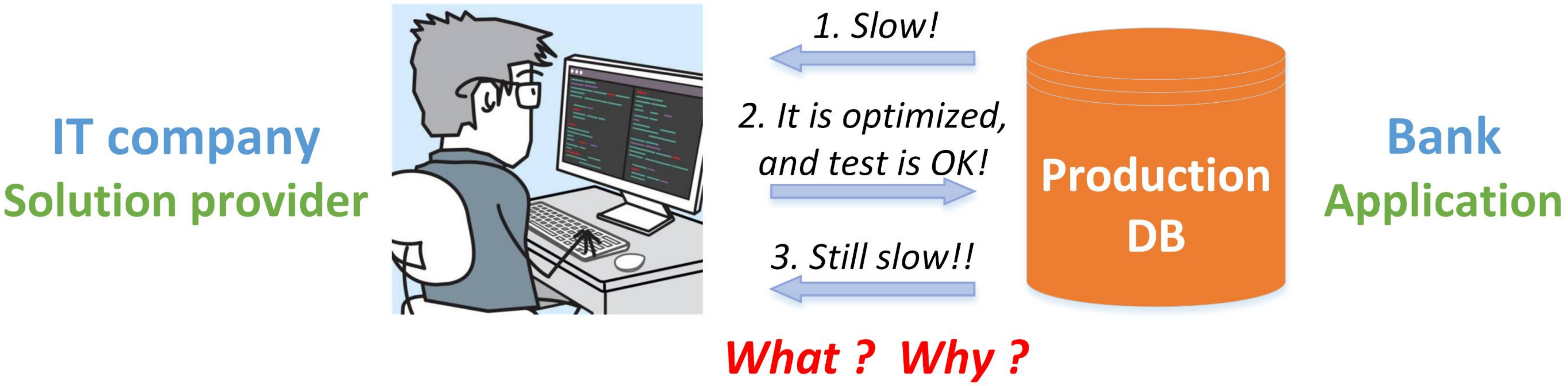
- Application scenarios: DBMS testing, database application testing, application-driven benchmarking.

# Test Databases Are Important!




- Application scenarios: DBMS testing, database application testing, application-driven benchmarking.

# Test Databases Are Important!

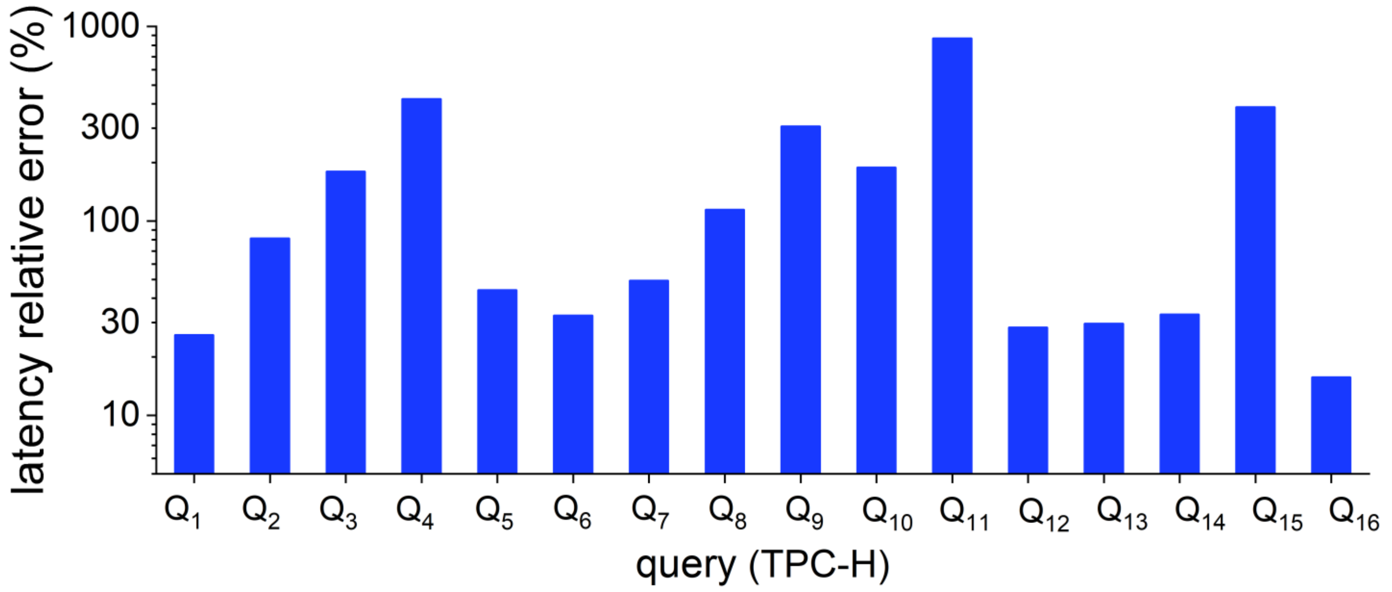



- Application scenarios: DBMS testing, database application testing, application-driven benchmarking.

# Random Test Database Is Deficient!

- The random test database has the **same** database schema and data characteristics as database generated by dbgen.
- There are **huge** execution cost **differences** between realistic database (dbgen) and synthetic database (random).

**Comparing the query latencies over database generated by dbgen and database randomly generated.**



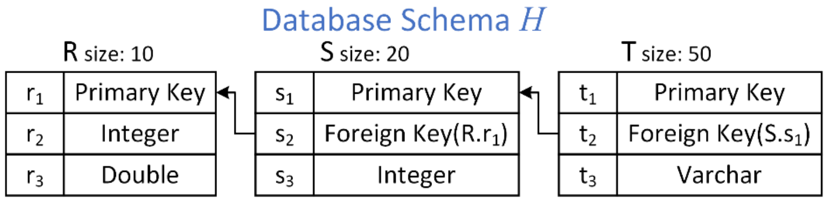

**The average relative error of query latencies is 175%!!**

# Query-Aware Data Generation

- **Input:** database schema, data characteristics and workload characteristics.
- **Output:** test database and instantiated query parameters.

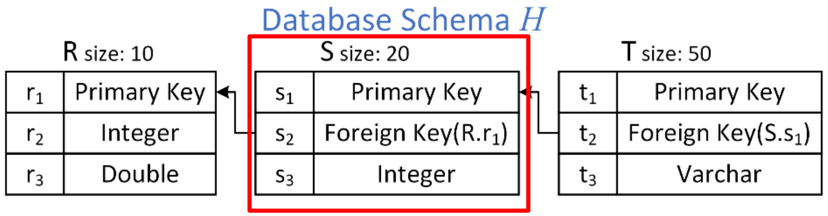
# Query-Aware Data Generation

- **Input:** database schema, data characteristics and workload characteristics.
- **Output:** test database and instantiated query parameters.



# Query-Aware Data Generation

- **Input:** database schema, data characteristics and workload characteristics.
- **Output:** test database and instantiated query parameters.



# Query-Aware Data Generation

- **Input:** database schema, data characteristics and workload characteristics.
- **Output:** test database and instantiated query parameters.

Database Schema $H$

R size: 10

| | |
|---|---|
| $r_1$ | Primary Key |
| $r_2$ | Integer |
| $r_3$ | Double |

S size: 20

| | |
|---|---|
| $s_1$ | Primary Key |
| $s_2$ | Foreign Key(R.$r_1$) |
| $s_3$ | Integer |

T size: 50

| | |
|---|---|
| $t_1$ | Primary Key |
| $t_2$ | Foreign Key(S.$s_1$) |
| $t_3$ | Varchar |

Data Characteristics $D$

| Attr | Null | Domain | Cardinality | Len(Avg,Max) |
|---|---|---|---|---|
| R.$r_2$ | 0 | [0, 10] | 5 | -- |
| R.$r_3$ | 0 | [0, 30] | -- | -- |
| S.$s_3$ | 0 | [0, 20] | 10 | -- |
| T.$t_3$ | 20% | -- | 8 | (20, 100) |

# Query-Aware Data Generation

- **Input:** database schema, data characteristics and workload characteristics.
- **Output:** test database and instantiated query parameters.

Database Schema $H$

R size: 10

| | |
|---|---|
| $r_1$ | Primary Key |
| $r_2$ | Integer |
| $r_3$ | Double |

S size: 20

| | |
|---|---|
| $s_1$ | Primary Key |
| $s_2$ | Foreign Key(R.$r_1$) |
| $s_3$ | Integer |

T size: 50

| | |
|---|---|
| $t_1$ | Primary Key |
| $t_2$ | Foreign Key(S.$s_1$) |
| $t_3$ | Varchar |

Data Characteristics $D$

| Attr | Null | Domain | Cardinality | Len(Avg,Max) |
|---|---|---|---|---|
| R.$r_2$ | 0 | [0, 10] | 5 | -- |
| R.$r_3$ | 0 | [0, 30] | -- | -- |
| S.$s_3$ | 0 | [0, 20] | 10 | -- |
| T.$t_3$ | 20% | -- | 8 | (20, 100) |

# Query-Aware Data Generation

- **Input:** database schema, data characteristics and workload characteristics.
- **Output:** test database and instantiated query parameters.

Database Schema $H$

R size: 10

| | |
|---|---|
| $r_1$ | Primary Key |
| $r_2$ | Integer |
| $r_3$ | Double |

S size: 20

| | |
|---|---|
| $s_1$ | Primary Key |
| $s_2$ | Foreign Key(R.$r_1$) |
| $s_3$ | Integer |

T size: 50

| | |
|---|---|
| $t_1$ | Primary Key |
| $t_2$ | Foreign Key(S.$s_1$) |
| $t_3$ | Varchar |

Data Characteristics $D$

| Attr | Null | Domain | Cardinality | Len(Avg,Max) |
|---|---|---|---|---|
| R.$r_2$ | 0 | [0, 10] | 5 | -- |
| R.$r_3$ | 0 | [0, 30] | -- | -- |
| S.$s_3$ | 0 | [0, 20] | 10 | -- |
| T.$t_3$ | 20% | -- | 8 | (20, 100) |

# Query-Aware Data Generation

- **Input:** database schema, data characteristics and workload characteristics.
- **Output:** test database and instantiated query parameters.

Database Schema $H$

R size: 10

| | |
|---|---|
| $r_1$ | Primary Key |
| $r_2$ | Integer |
| $r_3$ | Double |

S size: 20

| | |
|---|---|
| $s_1$ | Primary Key |
| $s_2$ | Foreign Key(R.$r_1$) |
| $s_3$ | Integer |

T size: 50

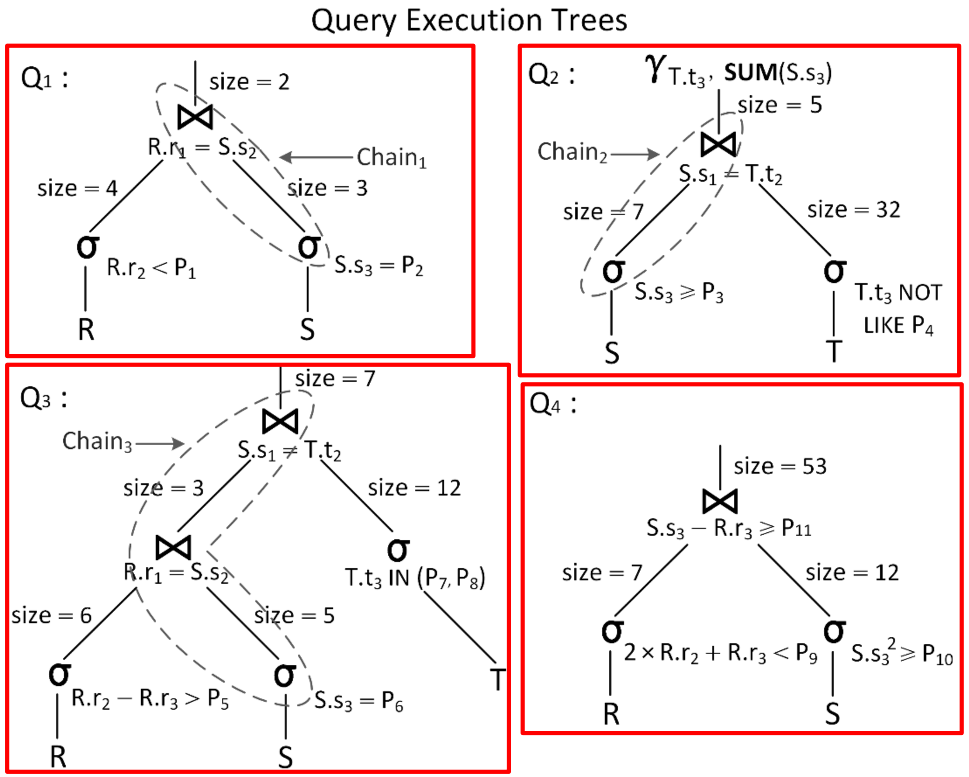
| | |
|---|---|
| $t_1$ | Primary Key |
| $t_2$ | Foreign Key(S.$s_1$) |
| $t_3$ | Varchar |

Data Characteristics $D$

| Attr | Null | Domain | Cardinality | Len(Avg,Max) |
|---|---|---|---|---|
| R.$r_2$ | 0 | [0, 10] | 5 | -- |
| R.$r_3$ | 0 | [0, 30] | -- | -- |
| S.$s_3$ | 0 | [0, 20] | 10 | -- |
| T.$t_3$ | 20% | -- | 8 | (20, 100) |

Workload Characteristics $W$

| | | | |
|---|---|---|---|
| $c_1$ | [$Q_1$, R.$r_2 < P_1$, 4] | $c_8$ | [$Q_3$, S.$s_3 = P_6$, 5] |
| $c_2$ | [$Q_1$, S.$s_3 = P_2$, 3] | $c_9$ | [$Q_3$, R.$r_1$ = S.$s_2$, 3] |
| $c_3$ | [$Q_1$, R.$r_1$ = S.$s_2$, 2] | $c_{10}$ | [$Q_3$, T.$t_3$ IN ($P_7$, $P_8$), 12] |
| $c_4$ | [$Q_2$, S.$s_3 \geqslant P_3$, 7] | $c_{11}$ | [$Q_3$, S.$s_1$ = T.$t_2$, 7] |
| $c_5$ | [$Q_2$, T.$t_3$ NOT LIKE $P_4$, 32] | $c_{12}$ | [$Q_4$, $2 \times$ R.$r_2$ + R.$r_3 < P_9$, 7] |
| $c_6$ | [$Q_2$, S.$s_1$ = T.$t_2$, 5] | $c_{13}$ | [$Q_4$, S.$s_3^2 \geqslant P_{10}$, 12] |
| $c_7$ | [$Q_3$, R.$r_2$ − R.$r_3 > P_5$, 6] | $c_{14}$ | [$Q_4$, S.$s_3$ − R.$r_3 \geqslant P_{11}$, 53] |




Parameterized queries

# Query-Aware Data Generation

- **Input:** database schema, data characteristics and workload characteristics.
- **Output:** test database and instantiated query parameters.

Database Schema $H$

R size: 10

| | |
|---|---|
| $r_1$ | Primary Key |
| $r_2$ | Integer |
| $r_3$ | Double |

S size: 20

| | |
|---|---|
| $s_1$ | Primary Key |
| $s_2$ | Foreign Key($R.r_1$) |
| $s_3$ | Integer |

T size: 50

| | |
|---|---|
| $t_1$ | Primary Key |
| $t_2$ | Foreign Key($S.s_1$) |
| $t_3$ | Varchar |

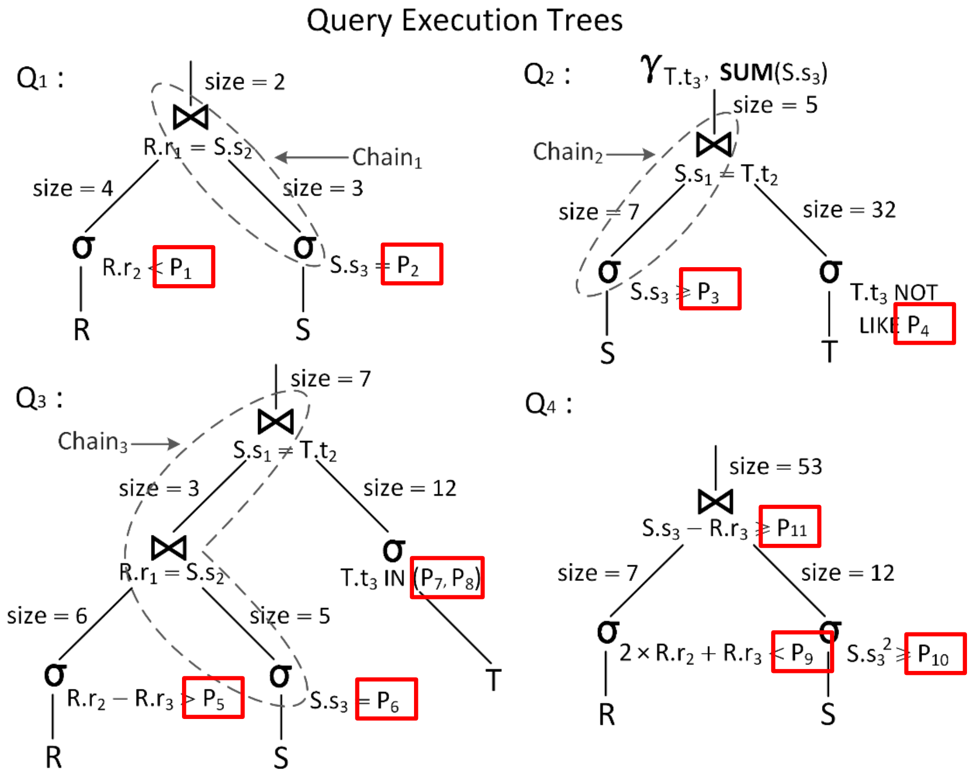
Data Characteristics $D$

| Attr | Null | Domain | Cardinality | Len(Avg,Max) |
|---|---|---|---|---|
| $R.r_2$ | 0 | [0, 10] | 5 | -- |
| $R.r_3$ | 0 | [0, 30] | -- | -- |
| $S.s_3$ | 0 | [0, 20] | 10 | -- |
| $T.t_3$ | 20% | -- | 8 | (20, 100) |

Workload Characteristics $W$

| | | | |
|---|---|---|---|
| $c_1$ | $[Q_1, R.r_2 < P_1, 4]$ | $c_8$ | $[Q_3, S.s_3 = P_6, 5]$ |
| $c_2$ | $[Q_1, S.s_3 = P_2, 3]$ | $c_9$ | $[Q_3, R.r_1 = S.s_2, 3]$ |
| $c_3$ | $[Q_1, R.r_1 = S.s_2, 2]$ | $c_{10}$ | $[Q_3, T.t_3$ IN $(P_7, P_8), 12]$ |
| $c_4$ | $[Q_2, S.s_3 \geqslant P_3, 7]$ | $c_{11}$ | $[Q_3, S.s_1 = T.t_2, 7]$ |
| $c_5$ | $[Q_2, T.t_3$ NOT LIKE $P_4, 32]$ | $c_{12}$ | $[Q_4, 2 \times R.r_2 + R.r_3 < P_9, 7]$ |
| $c_6$ | $[Q_2, S.s_1 = T.t_2, 5]$ | $c_{13}$ | $[Q_4, S.s_3^2 \geqslant P_{10}, 12]$ |
| $c_7$ | $[Q_3, R.r_2 - R.r_3 > P_5, 6]$ | $c_{14}$ | $[Q_4, S.s_3 - R.r_3 \geqslant P_{11}, 53]$ |

Query Execution Trees




Variable parameters

# Query-Aware Data Generation

- **Input:** database schema, data characteristics and workload characteristics.
- **Output:** test database and instantiated query parameters.

Database Schema $H$

R size: 10

| | |
|---|---|
| $r_1$ | Primary Key |
| $r_2$ | Integer |
| $r_3$ | Double |

S size: 20

| | |
|---|---|
| $s_1$ | Primary Key |
| $s_2$ | Foreign Key($R.r_1$) |
| $s_3$ | Integer |

T size: 50

| | |
|---|---|
| $t_1$ | Primary Key |
| $t_2$ | Foreign Key($S.s_1$) |
| $t_3$ | Varchar |

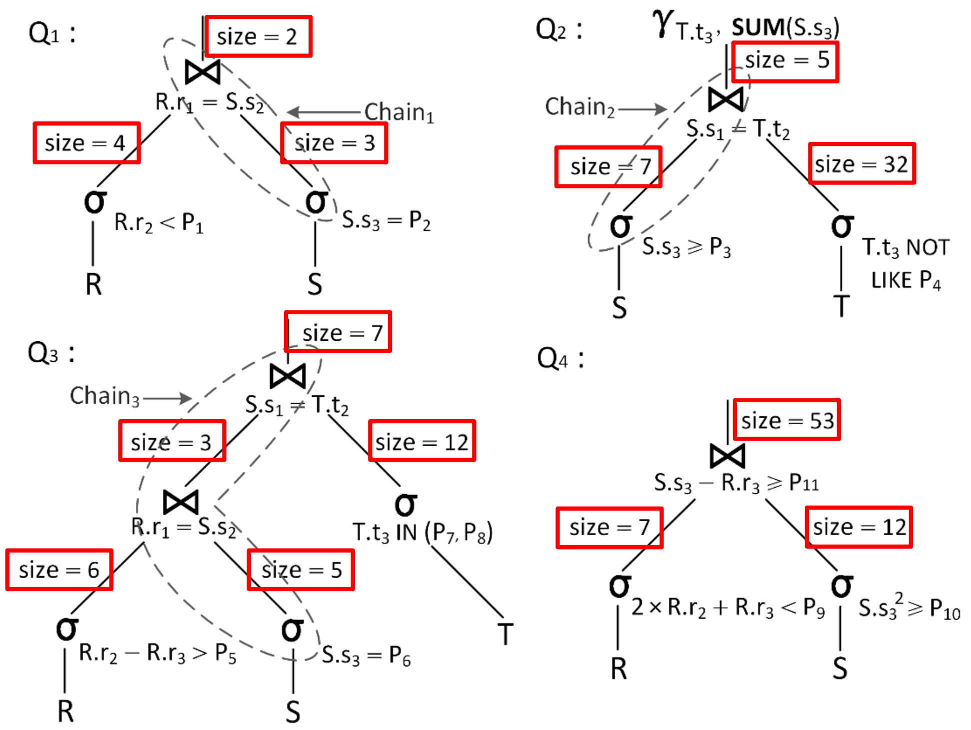
Data Characteristics $D$

| Attr | Null | Domain | Cardinality | Len(Avg,Max) |
|---|---|---|---|---|
| $R.r_2$ | 0 | [0, 10] | 5 | -- |
| $R.r_3$ | 0 | [0, 30] | -- | -- |
| $S.s_3$ | 0 | [0, 20] | 10 | -- |
| $T.t_3$ | 20% | -- | 8 | (20, 100) |

Workload Characteristics $W$

| | | | |
|---|---|---|---|
| $c_1$ | $[Q_1, R.r_2 < P_1, 4]$ | $c_8$ | $[Q_3, S.s_3 = P_6, 5]$ |
| $c_2$ | $[Q_1, S.s_3 = P_2, 3]$ | $c_9$ | $[Q_3, R.r_1 = S.s_2, 3]$ |
| $c_3$ | $[Q_1, R.r_1 = S.s_2, 2]$ | $c_{10}$ | $[Q_3, T.t_3$ IN $(P_7, P_8), 12]$ |
| $c_4$ | $[Q_2, S.s_3 \geqslant P_3, 7]$ | $c_{11}$ | $[Q_3, S.s_1 = T.t_2, 7]$ |
| $c_5$ | $[Q_2, T.t_3$ NOT LIKE $P_4, 32]$ | $c_{12}$ | $[Q_4, 2 \times R.r_2 + R.r_3 < P_9, 7]$ |
| $c_6$ | $[Q_2, S.s_1 = T.t_2, 5]$ | $c_{13}$ | $[Q_4, S.s_3^2 \geqslant P_{10}, 12]$ |
| $c_7$ | $[Q_3, R.r_2 - R.r_3 > P_5, 6]$ | $c_{14}$ | $[Q_4, S.s_3 - R.r_3 \geqslant P_{11}, 53]$ |

Query Execution Trees




Cardinality constraints

# Query-Aware Data Generation

- **Input:** database schema, data characteristics and workload characteristics.
- **Output:** test database and instantiated query parameters.

Database Schema $H$

R size: 10

| | |
|---|---|
| $r_1$ | Primary Key |
| $r_2$ | Integer |
| $r_3$ | Double |

S size: 20

| | |
|---|---|
| $s_1$ | Primary Key |
| $s_2$ | Foreign Key($R.r_1$) |
| $s_3$ | Integer |

T size: 50

| | |
|---|---|
| $t_1$ | Primary Key |
| $t_2$ | Foreign Key($S.s_1$) |
| $t_3$ | Varchar |

Data Characteristics $D$

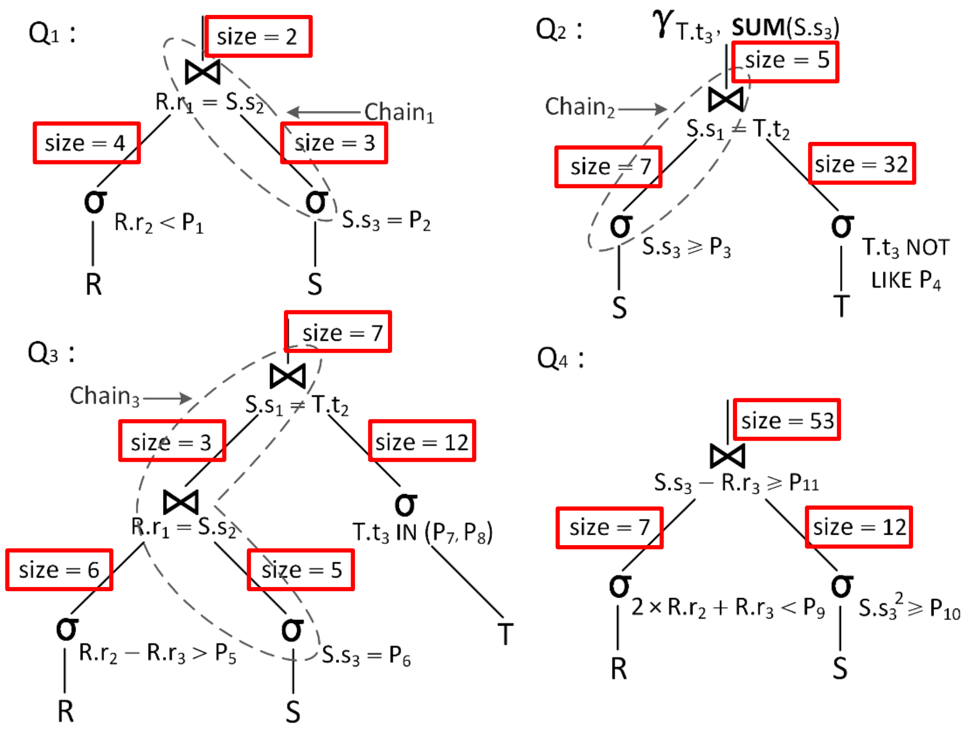
| Attr | Null | Domain | Cardinality | Len(Avg,Max) |
|---|---|---|---|---|
| $R.r_2$ | 0 | [0, 10] | 5 | -- |
| $R.r_3$ | 0 | [0, 30] | -- | -- |
| $S.s_3$ | 0 | [0, 20] | 10 | -- |
| $T.t_3$ | 20% | -- | 8 | (20, 100) |

All 14 cardinality constraints

Workload Characteristics $W$

| | | | |
|---|---|---|---|
| $c_1$ | $[Q_1, R.r_2 < P_1, 4]$ | $c_8$ | $[Q_3, S.s_3 = P_6, 5]$ |
| $c_2$ | $[Q_1, S.s_3 = P_2, 3]$ | $c_9$ | $[Q_3, R.r_1 = S.s_2, 3]$ |
| $c_3$ | $[Q_1, R.r_1 = S.s_2, 2]$ | $c_{10}$ | $[Q_3, T.t_3 \text{ IN } (P_7, P_8), 12]$ |
| $c_4$ | $[Q_2, S.s_3 \geqslant P_3, 7]$ | $c_{11}$ | $[Q_3, S.s_1 = T.t_2, 7]$ |
| $c_5$ | $[Q_2, T.t_3 \text{ NOT LIKE } P_4, 32]$ | $c_{12}$ | $[Q_4, 2 \times R.r_2 + R.r_3 < P_9, 7]$ |
| $c_6$ | $[Q_2, S.s_1 = T.t_2, 5]$ | $c_{13}$ | $[Q_4, S.s_3^2 \geqslant P_{10}, 12]$ |
| $c_7$ | $[Q_3, R.r_2 - R.r_3 > P_5, 6]$ | $c_{14}$ | $[Q_4, S.s_3 - R.r_3 \geqslant P_{11}, 53]$ |

Query Execution Trees



# Query-Aware Data Generation

- **Input:** database schema, data characteristics and workload characteristics.
- **Output:** test database and instantiated query parameters.

Database Schema $H$

R size: 10

| | |
|---|---|
| $r_1$ | Primary Key |
| $r_2$ | Integer |
| $r_3$ | Double |

S size: 20

| | |
|---|---|
| $s_1$ | Primary Key |
| $s_2$ | Foreign Key(R.$r_1$) |
| $s_3$ | Integer |

T size: 50

| | |
|---|---|
| $t_1$ | Primary Key |
| $t_2$ | Foreign Key(S.$s_1$) |
| $t_3$ | Varchar |

Data Characteristics $D$

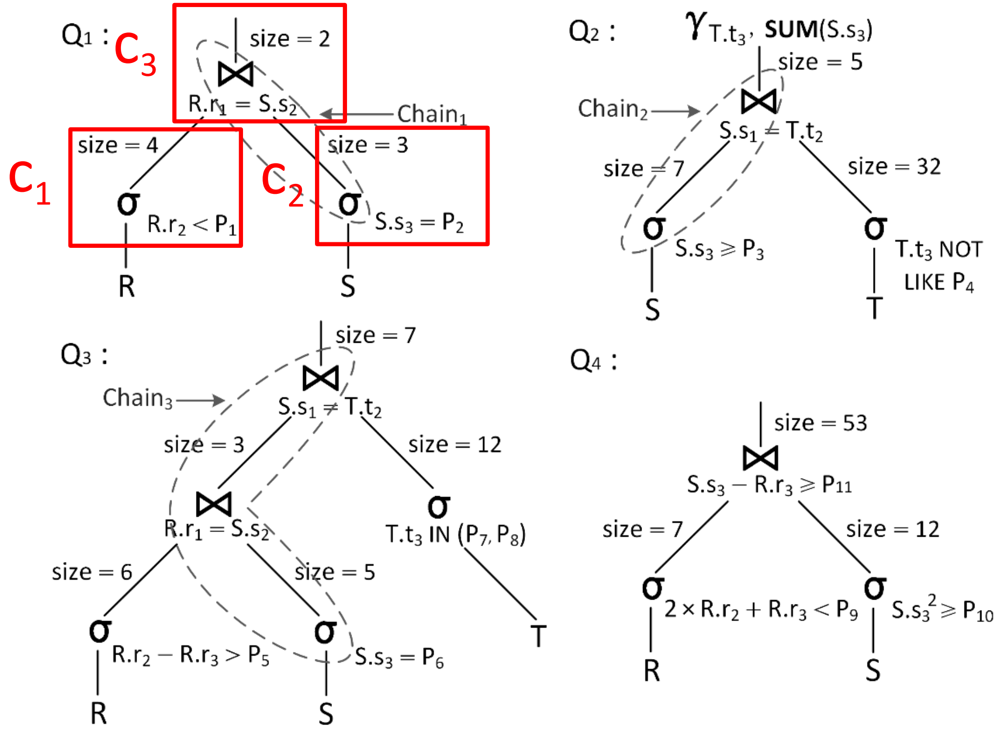
| Attr | Null | Domain | Cardinality | Len(Avg,Max) |
|---|---|---|---|---|
| R.$r_2$ | 0 | [0, 10] | 5 | -- |
| R.$r_3$ | 0 | [0, 30] | -- | -- |
| S.$s_3$ | 0 | [0, 20] | 10 | -- |
| T.$t_3$ | 20% | -- | 8 | (20, 100) |

Workload Characteristics $W$

Cardinality constraints on $Q_1$

| | | | |
|---|---|---|---|
| $c_1$ | [$Q_1$, R.$r_2 < P_1$, 4] | $c_8$ | [$Q_3$, S.$s_3 = P_6$, 5] |
| $c_2$ | [$Q_1$, S.$s_3 = P_2$, 3] | $c_9$ | [$Q_3$, R.$r_1$ = S.$s_2$, 3] |
| $c_3$ | [$Q_1$, R.$r_1$ = S.$s_2$, 2] | $c_{10}$ | [$Q_3$, T.$t_3$ IN ($P_7$, $P_8$), 12] |
| $c_4$ | [$Q_2$, S.$s_3 \geqslant P_3$, 7] | $c_{11}$ | [$Q_3$, S.$s_1$ = T.$t_2$, 7] |
| $c_5$ | [$Q_2$, T.$t_3$ NOT LIKE $P_4$, 32] | $c_{12}$ | [$Q_4$, $2 \times$ R.$r_2$ + R.$r_3 < P_9$, 7] |
| $c_6$ | [$Q_2$, S.$s_1$ = T.$t_2$, 5] | $c_{13}$ | [$Q_4$, S.$s_3^2 \geqslant P_{10}$, 12] |
| $c_7$ | [$Q_3$, R.$r_2$ − R.$r_3 > P_5$, 6] | $c_{14}$ | [$Q_4$, S.$s_3$ − R.$r_3 \geqslant P_{11}$, 53] |

Query Execution Trees



# Query-Aware Data Generation

- **Input:** database schema, data characteristics and workload characteristics.
- **Output:** test database and instantiated query parameters.

Database Schema $H$

R size: 10

| | |
|---|---|
| $r_1$ | Primary Key |
| $r_2$ | Integer |
| $r_3$ | Double |

S size: 20

| | |
|---|---|
| $s_1$ | Primary Key |
| $s_2$ | Foreign Key($R.r_1$) |
| $s_3$ | Integer |

T size: 50

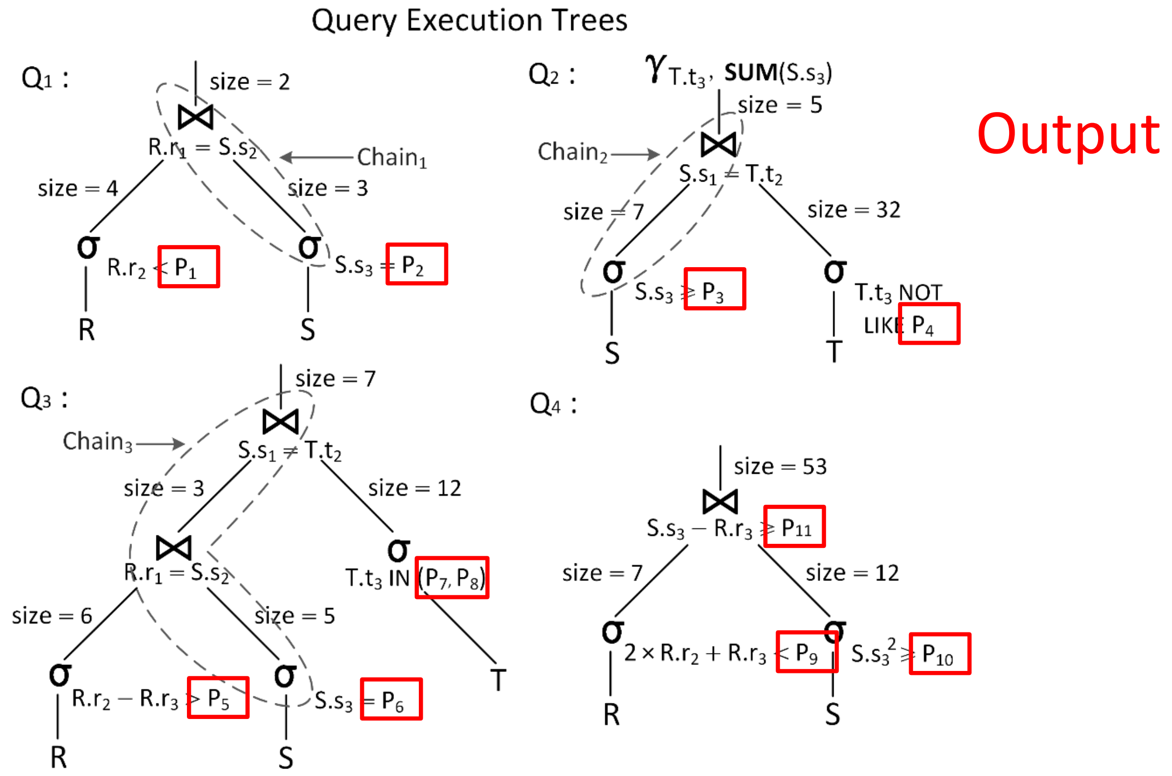
| | |
|---|---|
| $t_1$ | Primary Key |
| $t_2$ | Foreign Key($S.s_1$) |
| $t_3$ | Varchar |

Data Characteristics $D$

| Attr | Null | Domain | Cardinality | Len(Avg,Max) |
|---|---|---|---|---|
| $R.r_2$ | 0 | [0, 10] | 5 | -- |
| $R.r_3$ | 0 | [0, 30] | -- | -- |
| $S.s_3$ | 0 | [0, 20] | 10 | -- |
| $T.t_3$ | 20% | -- | 8 | (20, 100) |

Workload Characteristics $W$

| | | | |
|---|---|---|---|
| $c_1$ | $[Q_1, R.r_2 < P_1, 4]$ | $c_8$ | $[Q_3, S.s_3 = P_6, 5]$ |
| $c_2$ | $[Q_1, S.s_3 = P_2, 3]$ | $c_9$ | $[Q_3, R.r_1 = S.s_2, 3]$ |
| $c_3$ | $[Q_1, R.r_1 = S.s_2, 2]$ | $c_{10}$ | $[Q_3, T.t_3 \text{ IN } (P_7, P_8), 12]$ |
| $c_4$ | $[Q_2, S.s_3 \geqslant P_3, 7]$ | $c_{11}$ | $[Q_3, S.s_1 = T.t_2, 7]$ |
| $c_5$ | $[Q_2, T.t_3 \text{ NOT LIKE } P_4, 32]$ | $c_{12}$ | $[Q_4, 2 \times R.r_2 + R.r_3 < P_9, 7]$ |
| $c_6$ | $[Q_2, S.s_1 = T.t_2, 5]$ | $c_{13}$ | $[Q_4, S.s_3^2 \geqslant P_{10}, 12]$ |
| $c_7$ | $[Q_3, R.r_2 - R.r_3 > P_5, 6]$ | $c_{14}$ | $[Q_4, S.s_3 - R.r_3 \geqslant P_{11}, 53]$ |



# Comparison to Related Works

- The performance of state-of-the-art solutions remains far from satisfactory.

| | QAGen *SIGMOD 2007* | WAGen *VLDB 2010* | DCGen *SIGMOD 2011* | MyBenchmark *VLDBJ 2014* | Touchstone *ATC 2018* |
|---|---|---|---|---|---|
| Full Parallelization | No | No | Yes | No | Yes |
| Linear Scalability | No | No | No | No | Yes |
| Austere Mem Consumption | No | No | No | No | Yes |
| Wide Workload Support | No | No | No | No | Yes |
| Minimal Human Effort | Yes | Yes | No | Yes | Yes |

Touchstone is the **first** query-aware data generator which can support **full parallel** data generation on **multiple nodes**. And Touchstone is capable of supporting **industrial scale** database generation.

# Why do Previous Studies not Work?

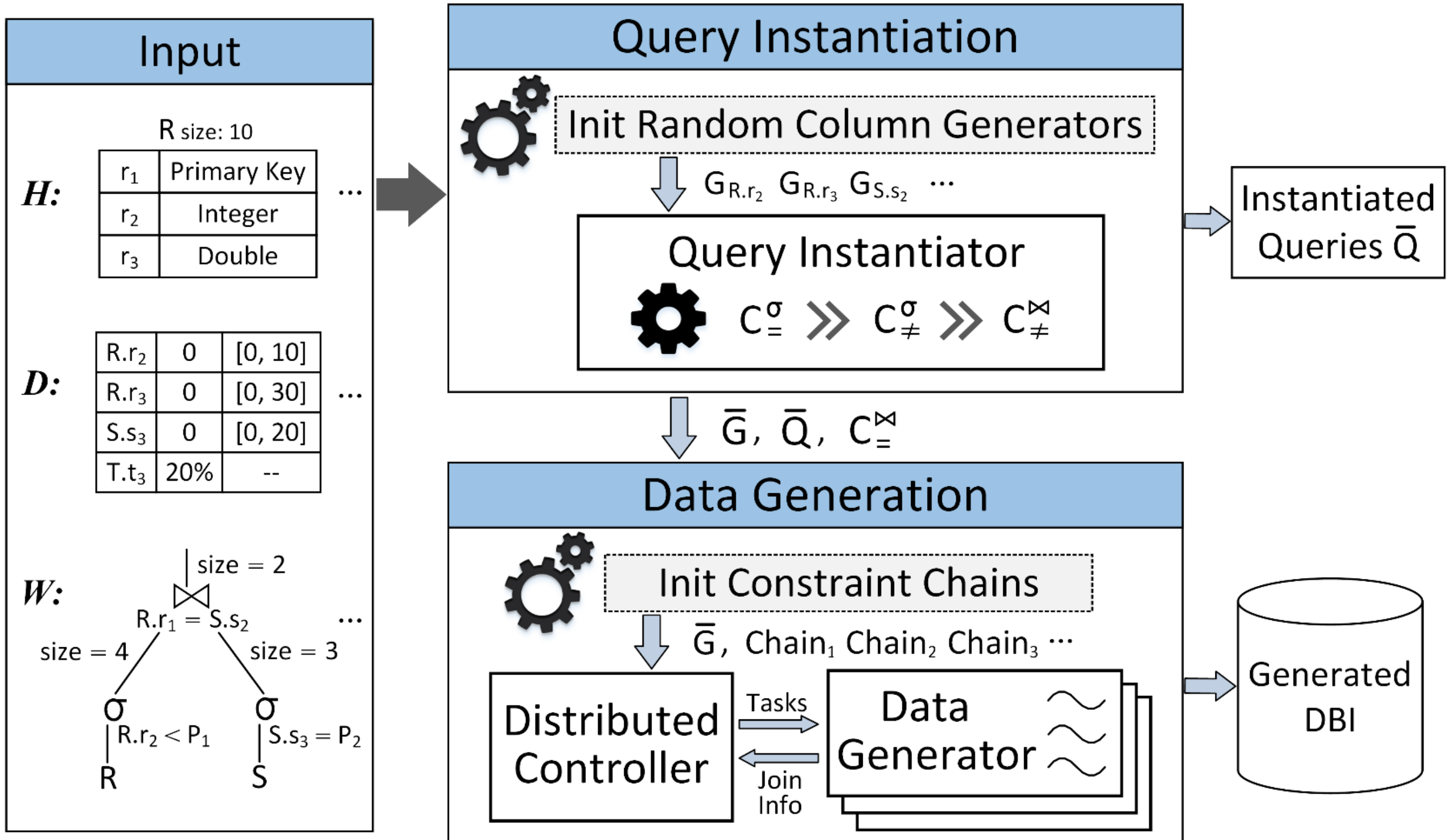
- **Primitive data generation algorithm**
  - Can not support fully parallel data generation in a distributed environment;
  - Can not support the non-equi-join workload.
- **Huge intermediate state dataset**
  - The memory consumption strongly depends on the size of generation outputs;
  - Is not scalable in generation database size.

# How does Touchstone Solve These Problems?

- **New query instantiation scheme**
  - Algorithms: binary search, random sampling;
  - Function: instantiating all the **variable parameters**.
- **New data generation scheme**
  - Algorithms: data generation using constraint chains, data compression on join information table;
  - Function: facilitating **parallel data generation** on multiple nodes with **austere memory consumption**.

# Overall Architecture

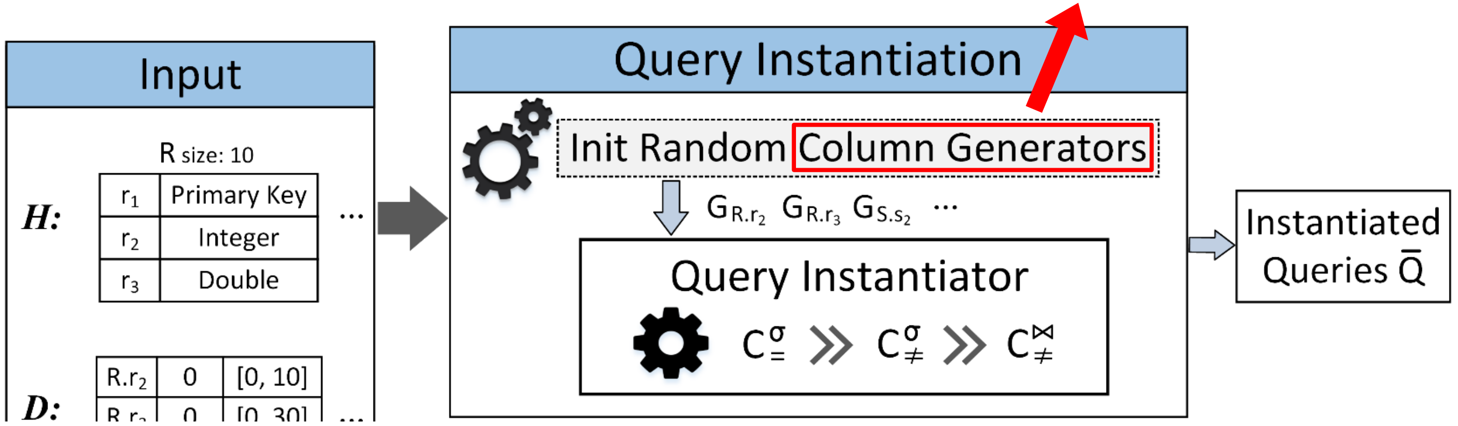
- **Query instantiation**
  - Initialize random column generators;
  - Instantiate symbolic query parameters.
- **Data generation**
  - Decompose the query trees annotated with cardinality constraints into constraint chains;
  - Generate data in parallel on multiple nodes.



# Overall Architecture

- **Query instantiation**
  - Initialize random column generators;
  - Instantiate symbolic query parameters.
- **Data generation**
  - Decompose the query trees annotated with cardinality constraints into constraint chains;
  - Generate data in parallel ( multiple nodes.

Determine the data distribution of columns




**Equality constraints over filters**

$S.s_3 = P_2$

$T.t_3$ NOT LIKE $P_4$

**Non-Equality constraints over filters**

$R.r_2 < P_1$

$R.r_2 - R.r_3 > P_5$

**Non-Equality constraints over joins**

$S.s_3 - R.r_3 \geq P_{11}$

# Overall Architecture

- Query instantiation
  - Initialize random column generators;
  - Instantiate symbolic query parameters.
- **Data generation**
  - Decompose the query trees annotated with cardinality constraints into constraint chains;
  - Generate data in parallel on multiple nodes.

**Compilation Step**

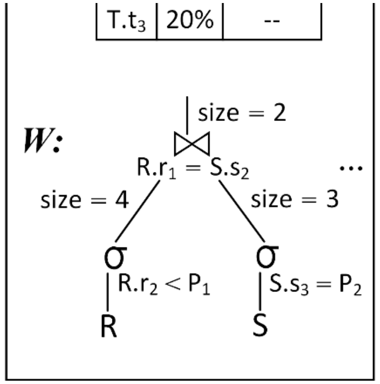
**(1) Order the tables as a generation sequence**

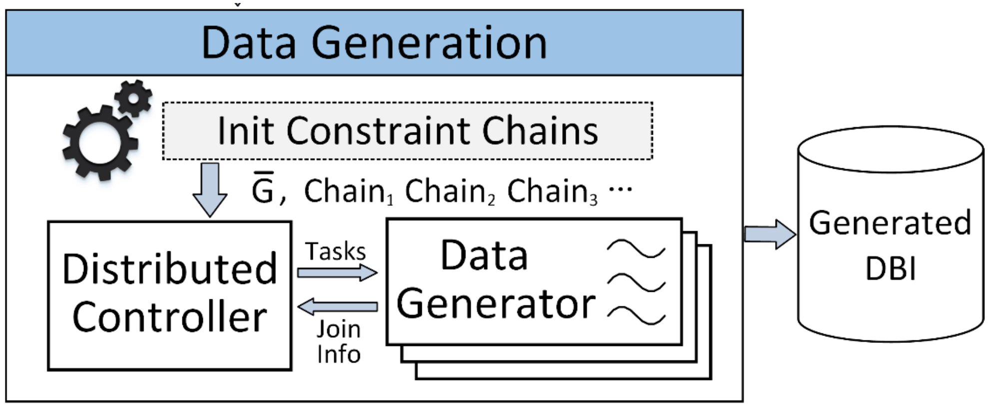
**(2) Decompose the query trees into constraint chains**

**Assembling Step**

**(1) Generate tuples in parallel on multiple nodes**

**(2) Efficiently manage the join information**

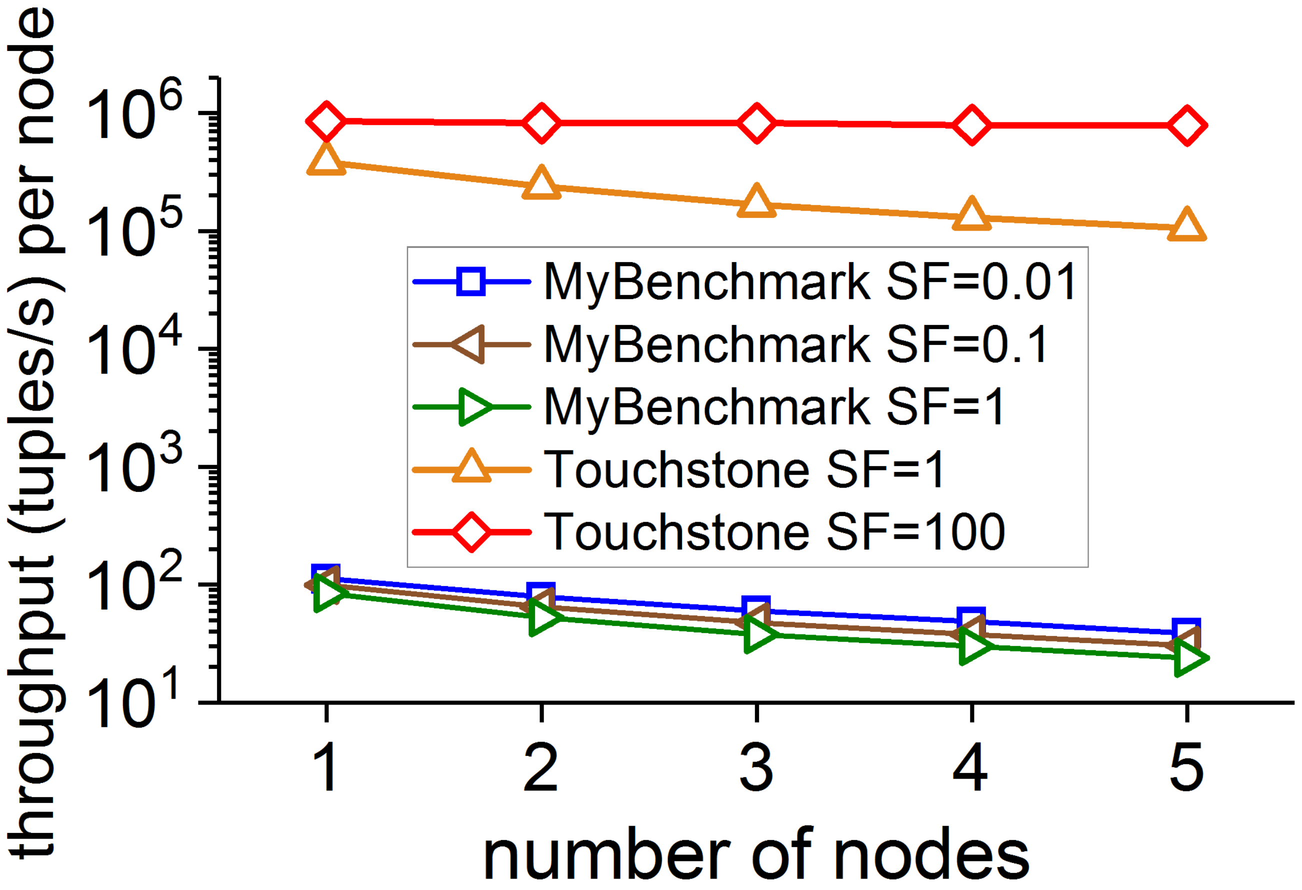


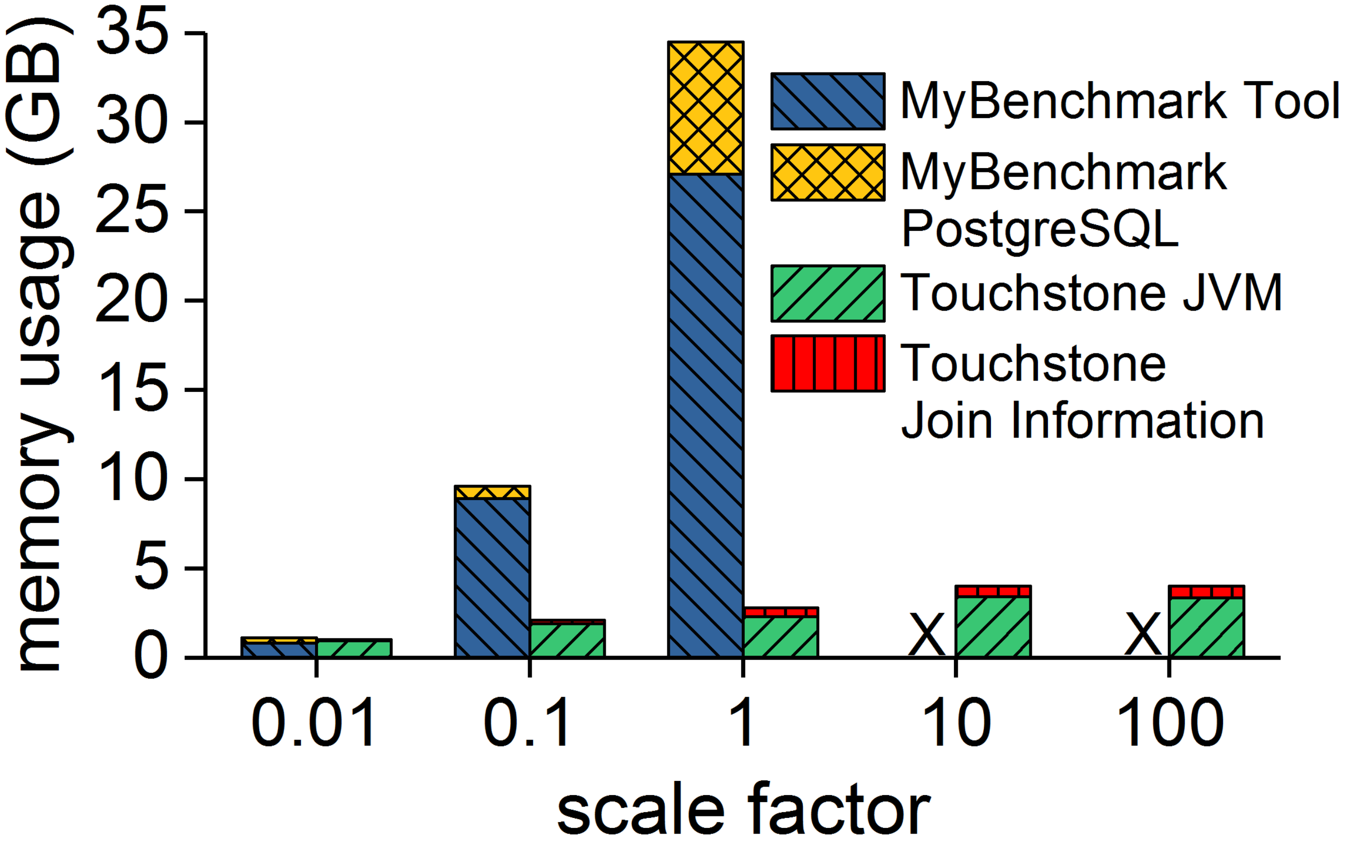
# Experiments

- **Test environment**
  - Cluster: 8 nodes
  - CPU: 2 * Intel Xeon E5-2620 @ 2.0 GHz
  - DRAM: 64GB
  - Disk: 3TB HDD configured in RAID-5
  - Network: 1 Gigabit Ethernet
- **Test workloads**
  - TPC-H benchmark (the first 16 queries) & Star schema benchmark (all 13 queries)
- **Comparison**
  - MyBenchmark [VLDBJ 2014]

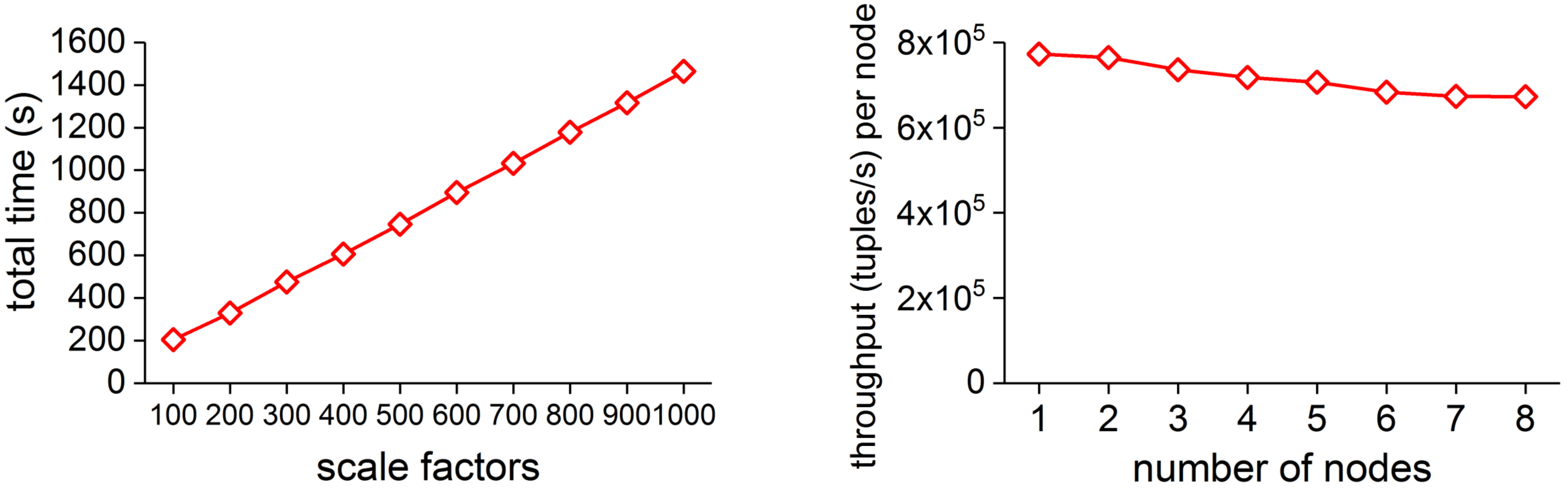
# Touchstone outperforms MyBenchmark on data generation throughput by orders



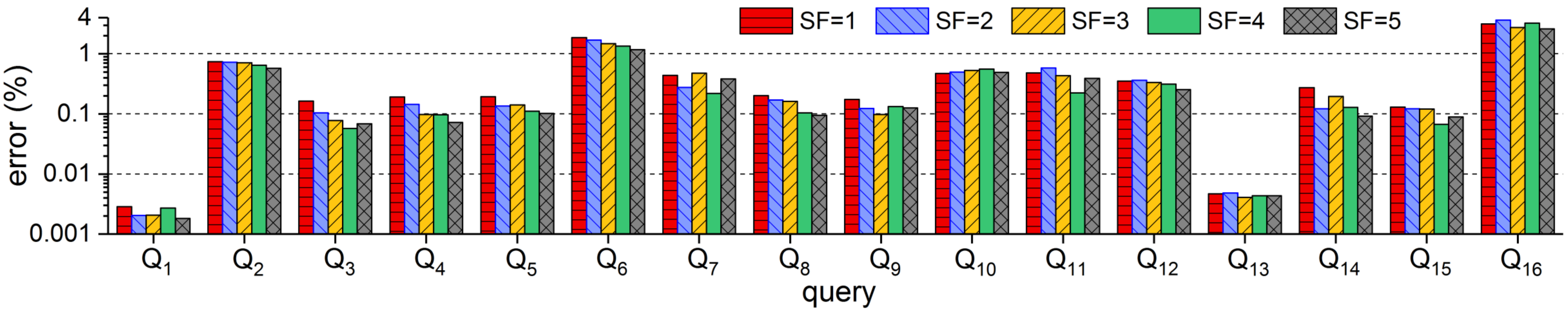
# The memory consumption of Touchstone is minimal

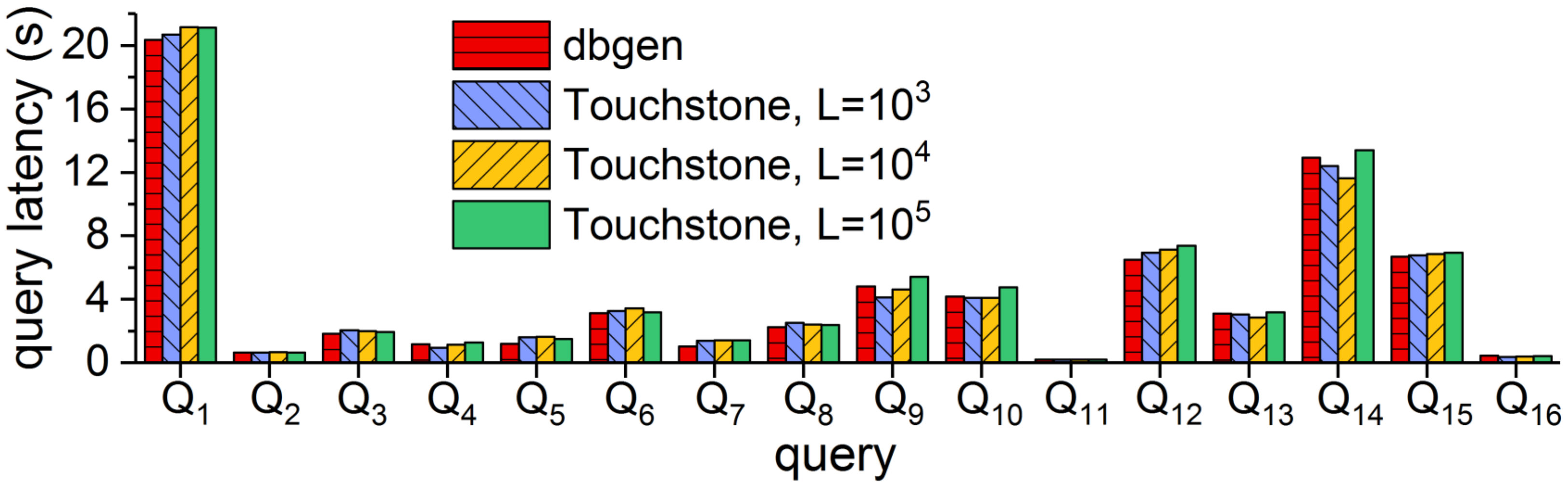


# Touchstone has linear scalability



# The workload on synthetic database matches the expectation on result cardinality and query latency



# Limitations & Conclusion

- **Limitations:**
  - – Touchstone does not support filters on key columns;
  - – Equality constraints over filters involving multiple columns are not supported;
  - – Equi-joins on columns with no reference constraint are not supported;
  - – Touchstone does not support the database schema with cyclic reference relationship.
- **Conclusion:**
  - – Touchstone is a query-aware data generator with characteristics of **completely parallelizable** and **bounded usage to memory**. And Touchstone is **linearly scalable to computing resource and data scale**.

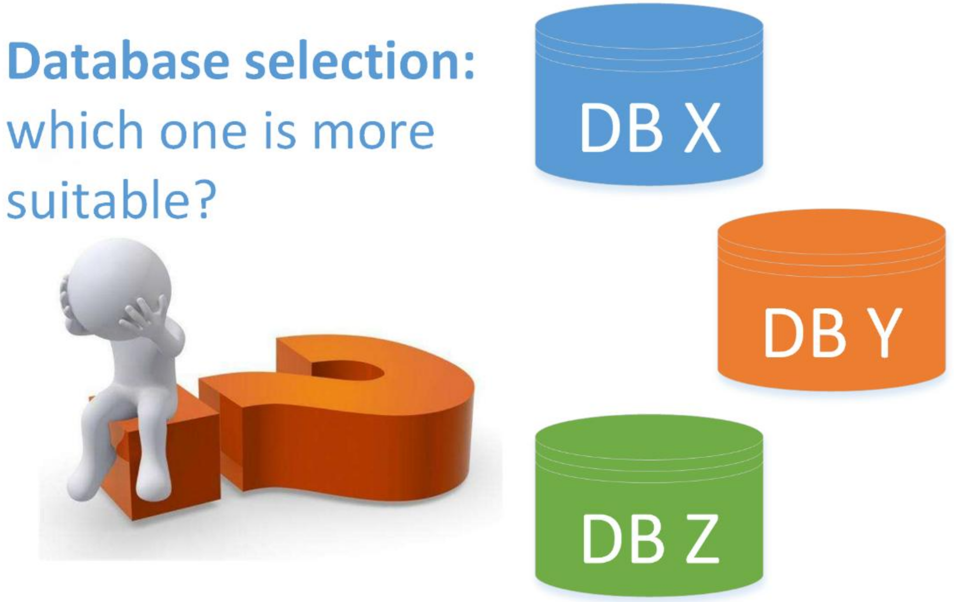
Thank you!!

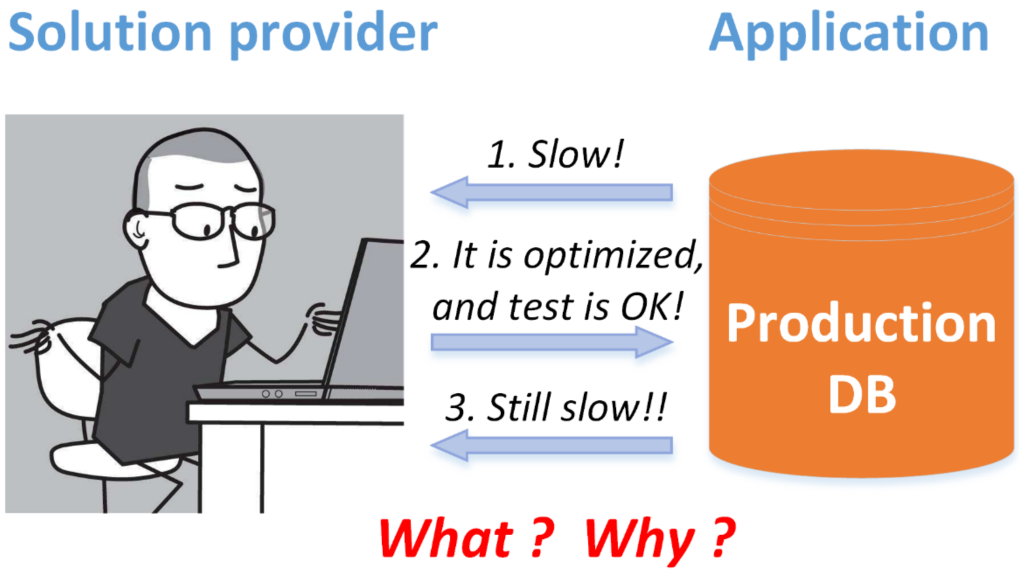
Q & A

https://github.com/daseECNU/Touchstone.

# Test Databases Are Important!

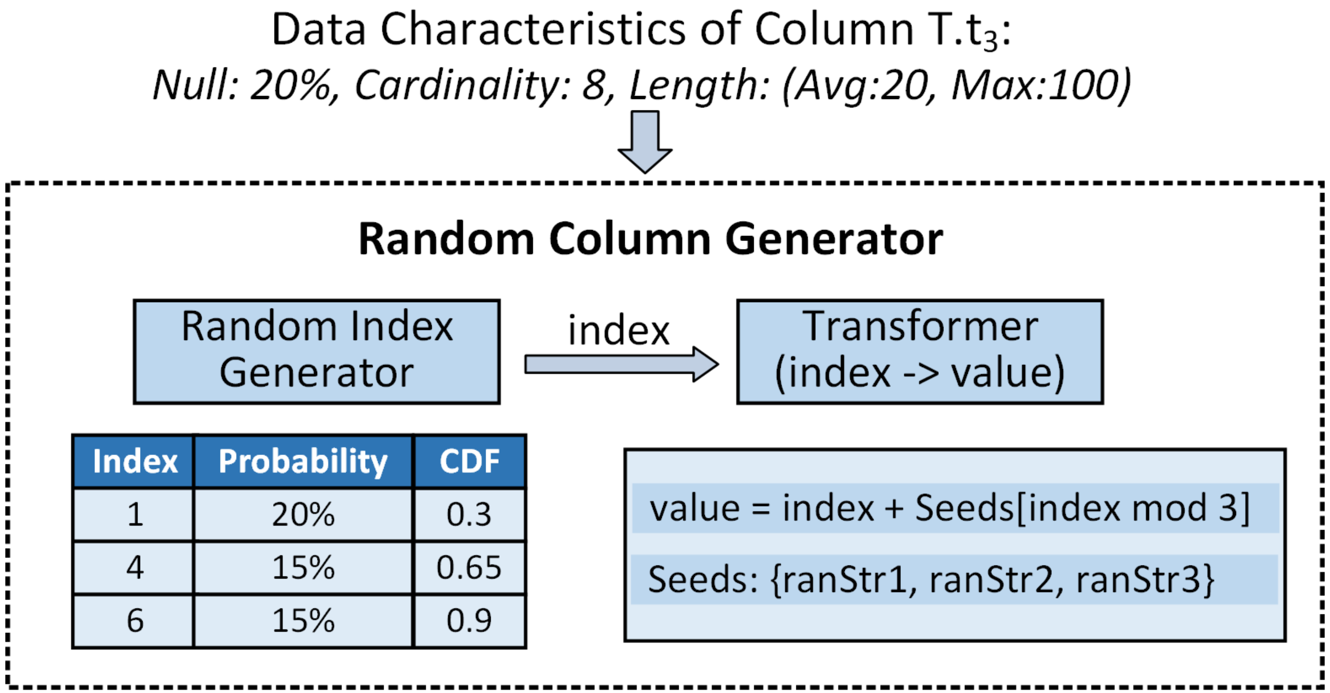
- Applications: DBMS testing, database application testing, application-driven benchmarking.



# Random Column Generator

- **Random index generator** outputs indexes from 0 to n-1 while n is the specified **cardinality**, and manipulates the **data distribution** of column values.
- **Index2Value transformer** **deterministically** maps the index to a **concrete value** in the specified domain of the column.




| Index | Probability | CDF |
|---|---|---|
| 1 | 20% | 0.3 |
| 4 | 15% | 0.65 |
| 6 | 15% | 0.9 |

# Query Instantiation

- The query instantiation is responsible for handling three types of cardinality constraints, i.e., $\boldsymbol{C^{\sigma}_{=}}$, $\boldsymbol{C^{\sigma}_{\neq}}$, $\boldsymbol{C^{\bowtie}_{\neq}}$. The fourth type of constraints $C^{\bowtie}_{=}$ is taken care of by the data generation process at runtime.

***This is an iterative process!***

**Equality constraints over filters**

$S.s_3 = P_2$

$T.t_3$ NOT LIKE $P_4$

**Assign a value to $P_2/P_4$ and adjust the data distribution**

→

**Non-Equality constraints over filters**

$R.r_2 < P_1$

$R.r_2 - R.r_3 > P_5$

**Search a value for $P_1/P_5$ based on the fixed data distribution**

→

**Non-Equality constraints over joins**

$S.s_3 - R.r_3 \geq P_{11}$

**Search a value for $P_{11}$ after settling the child nodes**

# Equality Constraints over Filters

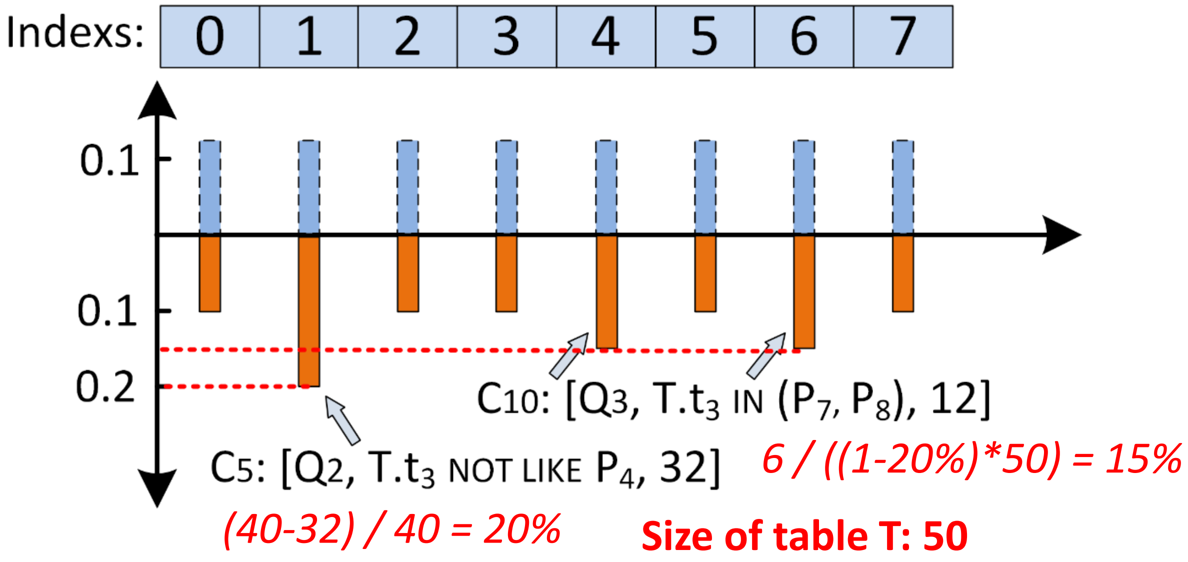
| Index | Probability | CDF |
|---|---|---|
| 1 | 20% | 0.3 |
| 4 | 15% | 0.65 |
| 6 | 15% | 0.9 |

- **(1)** Randomly select an index and obtain the corresponding value for instantiating the parameter;
- **(2)** Update the occurrence probability of the selected index in the column generator;
- **(3)** Calculate the cumulative probabilities in the probability table.

$\boldsymbol{D}$ of T.$t_3$: 20%, 8, (20, 100)

Value = *NULL* or (Index + Seeds[Index mod 3])

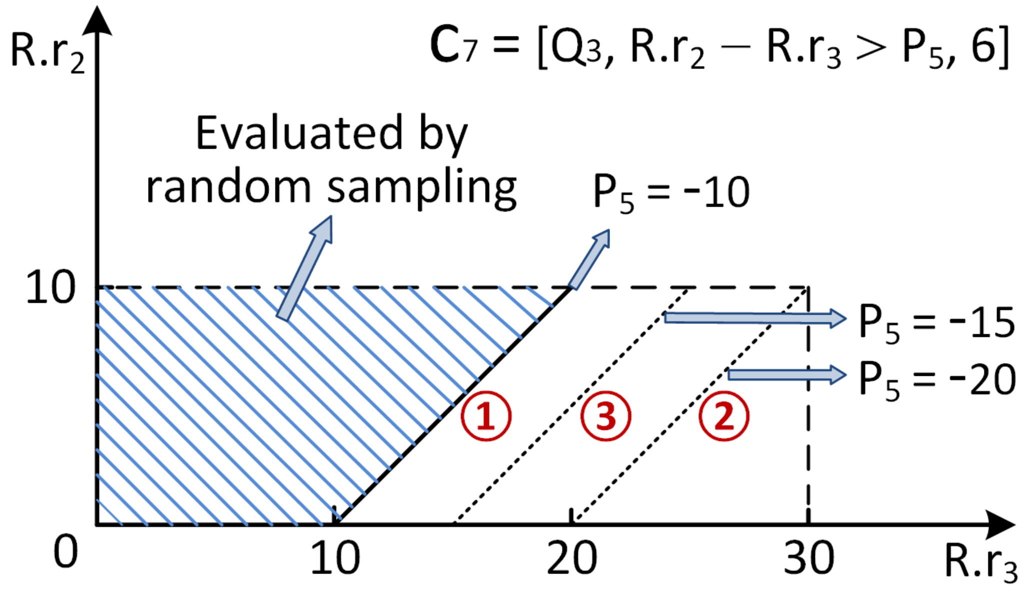
Seeds: {$ranStr_1$, $ranStr_2$, $ranStr_3$}



# Non-Equality Constraints over Filters

- Run a **binary search** over the parameter domain to find the optimal concrete parameter based on the fixed column data distribution.
- Using the **random sampling** algorithm to evaluate the probability of tuples satisfying the instantiated predicate.

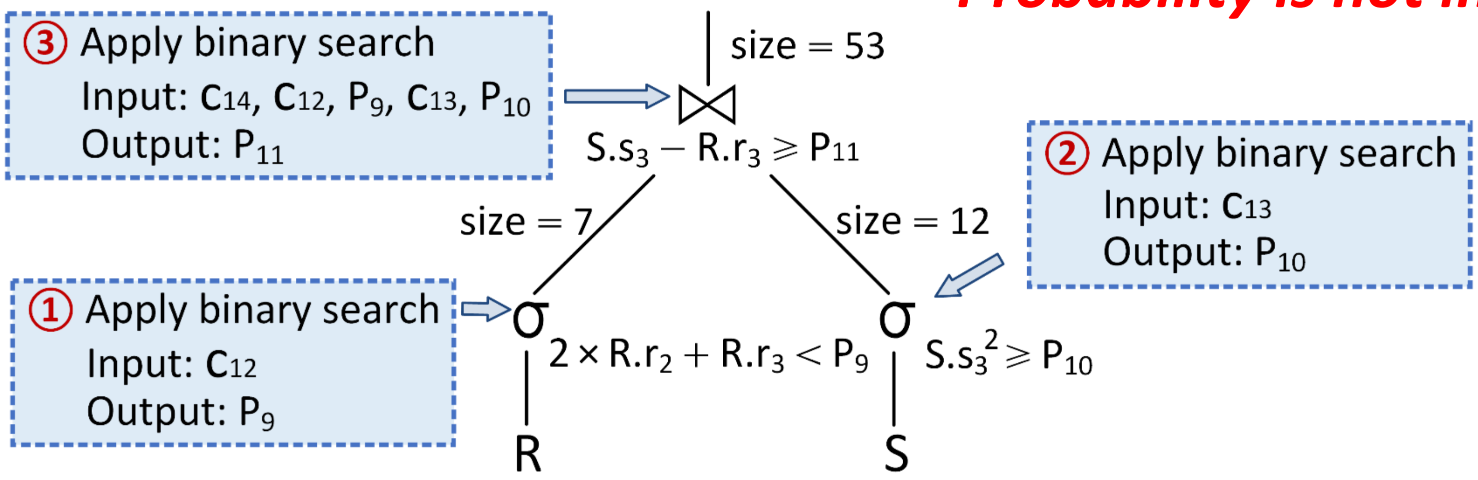
**Parameter searching procedure**



# Non-Equality Constraints over Joins

- We must process the constraints in a **bottom-up** manner, because the columns involved in constraints $C_{\neq}^{\bowtie}$ may overlap with the columns in the child nodes.

***Probability is not independent!***




*The processing strategy for each constraint in $C_{\neq}^{\bowtie}$ is the same as the constraint in $C_{\neq}^{\sigma}$ (binary search & random sampling).*

# Data Generation

- The data generation component is responsible for **assembling tuples** based on the outputs of the **column generators**.
- The key technical **challenge** here is to meet the equality constraints over the join operators, i.e., $C^{\bowtie}_{=}$, which involve **the dependencies among primary and foreign keys** from multiple tables.

**Compilation Step**

(1) Order the tables as a generation sequence

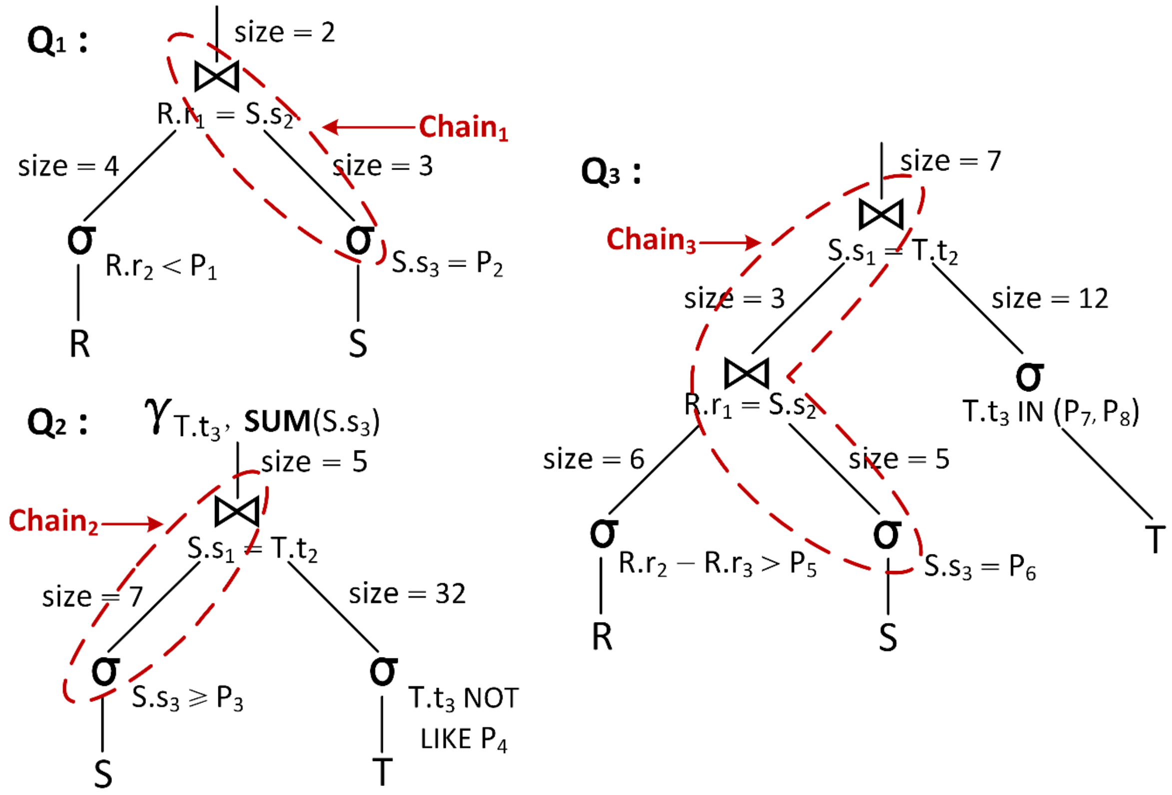
(2) Decompose the query trees into constraint chains

**Assembling Step**

(1) Generate tuples in parallel on multiple nodes

(2) Efficiently manage the join information

# Compilation Step




**Constraint chains of table R:**

Filter[$R.r_2 < P_1$] -> PK[$R.r_1$]

Filter[$R.r_2 - R.r_3 > P_5$] -> PK[$R.r_1$]

For $S.s_2$ referenced to $R.r_1$

**Constraint chains of table S:**

Filter[$S.s_3 = P_2$] -> FK[$S.s_2$, $R.r_1$, 2/3] ← **Chain$_1$**

Filter[$S.s_3 \geqslant P_3$] -> PK[$S.s_1$] ← **Chain$_2$**

Filter[$S.s_3 = P_6$] -> FK[$S.s_2$, $R.r_1$, 3/5] -> PK[$S.s_1$] ← **Chain$_3$**

For $T.t_2$ referenced to $S.s_1$

**Constraint chains of table T:**

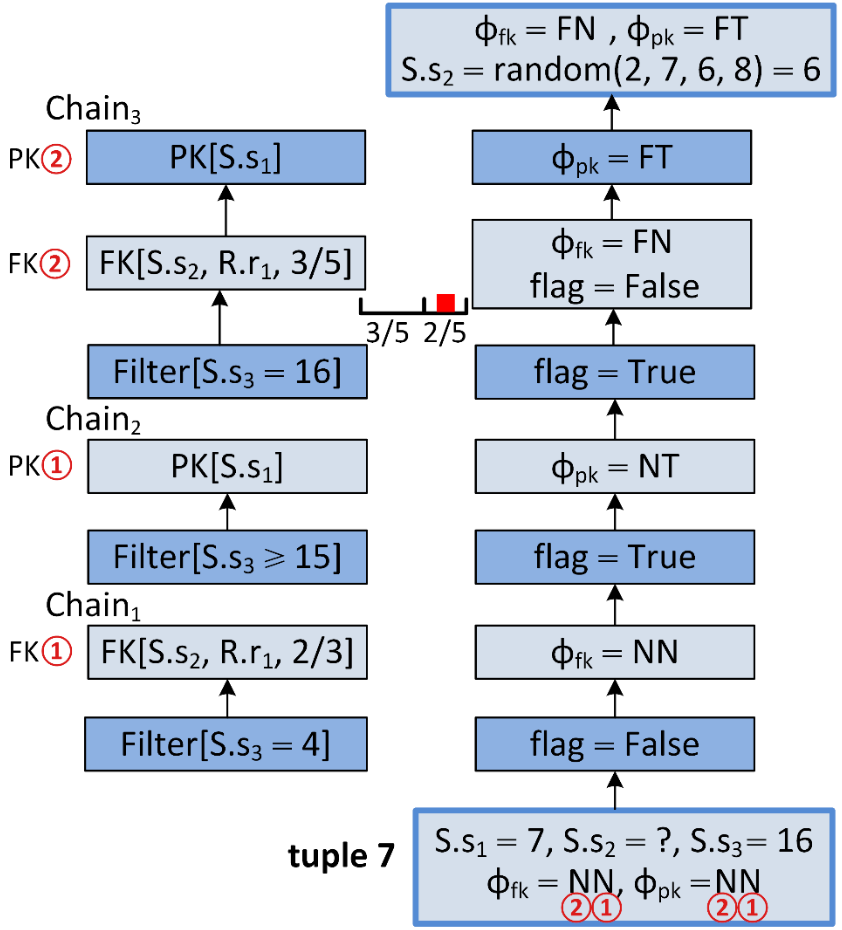
Filter[$T.t_3$ NOT LIKE $P_4$] -> FK[$T.t_2$, $S.s_1$, 5/32]

Filter[$T.t_3$ IN ($P_7$, $P_8$)] -> FK[$T.t_2$, $S.s_1$, 7/12]

**Focus on the manipulation of primary key and foreign keys.**

# Assembling Step

- **(1)** Incrementally assign a primary key;
- **(2)** Fill values in the non-key columns by calling the random column generators;
- **(3)** Identify the appropriate candidate referenced keys for each foreign key;
- **(4)** Maintain the join information of the primary key.




**Store up to *L* values!**

Join information table of $R.r_1$ ($\mathbf{t_{rpk}}$):

| bitmap | keys |
|---|---|
| TT | 3, 5 |
| TF | 0, 1, 4, 9 |
| FT | 2, 7 |
| FF | 6, 8 |

Join information table of $S.s_1$ ($\mathbf{t_{pk}}$):

| bitmap | keys |
|---|---|
| TT | 3, 5 |
| TF | -- |
| FT | 0, **7** |
| FF | 1, 2, 4, 6 |

# Handling Mismatch Cases

- There are some joinability statuses of the primary key that never occur!
- Therefore, in the tuple generation, it should be avoided to search such referenced primary key.
- The main idea is to add rules to manipulate relevant FK constraints.

Join information table of rpk:

| bitmap | keys |
|---|---|
| FFF | 1, 5 |
| TFF | 6, 7 |
| FFT | 2, 9 |
| T T F | 3, 8 |

③②①

The example constraint chains of the target table:

0.1

Filter[...] -> FK[fk,rpk,0.3]
0.35 ①
No adjustment

Filter[...] -> FK[fk,rpk,0.6] -> PK[...]
0.4 ②
FK[fk,rpk,0.65,rules:[FT <- T]]

③ FK[fk,rpk,0.2] -> Filter[...] -> PK[...]
FK[fk,rpk,0.17,rules:[FFT <- FT,TTF <- TF]]

***An example of adjustments to FK constraints***