# Reversing Statistics for Scalable Test Databases Generation

Entong Shen  Lyublena Antova

Pivotal (formerly Greenplum)

DBTest 2013, New York, June 24

Pivotal™

# Motivation

- Data generators: functional and performance testing, benchmarking, debugging customer issues, etc.
- Benchmarks (e.g.TPC-H,TPC-DS) often have different characteristics compared to real customer data
- Customer data usually hard to obtain
  - Sensitive and valuable info
  - Huge data volume
- Goal: a fast and practical data generator such that generated data close to real customer data
- Approach: generate data out of database metadata

# Related Work

- Limitations of benchmark data generators [e.g. `dbgen`, `dsgen`]
  - Fixed schema and workload
  - Limited control over generated dataset
- Workload-oblivious data generators
  - Closed-form column distributions [Gray et al. 1994]
  - Descriptive languages for data dependencies and column distributions [Bruno 2005, Hoag 2007, Rabl 2011]
- Workload-aware, constraint-based data generators [Torlak 2012, Arasu 2011, Binnig 2007, de la Riva 2010, Lo 2010]
  - Constraints specified by declarative languages
  - Common use of constraint solvers
  - May be expensive

# Design Goals

- Fast and practical
  - Generating large amounts of data in a short time
  - No solvers (linear programming, entropy maximization)
- Scalable
  - Support trillions of record
- Support strong constraints and some soft constraints
- Truthful
- Extensible
  - New source of input metadata
  - Evolvement of database statistic model
- No language or data profiler needed

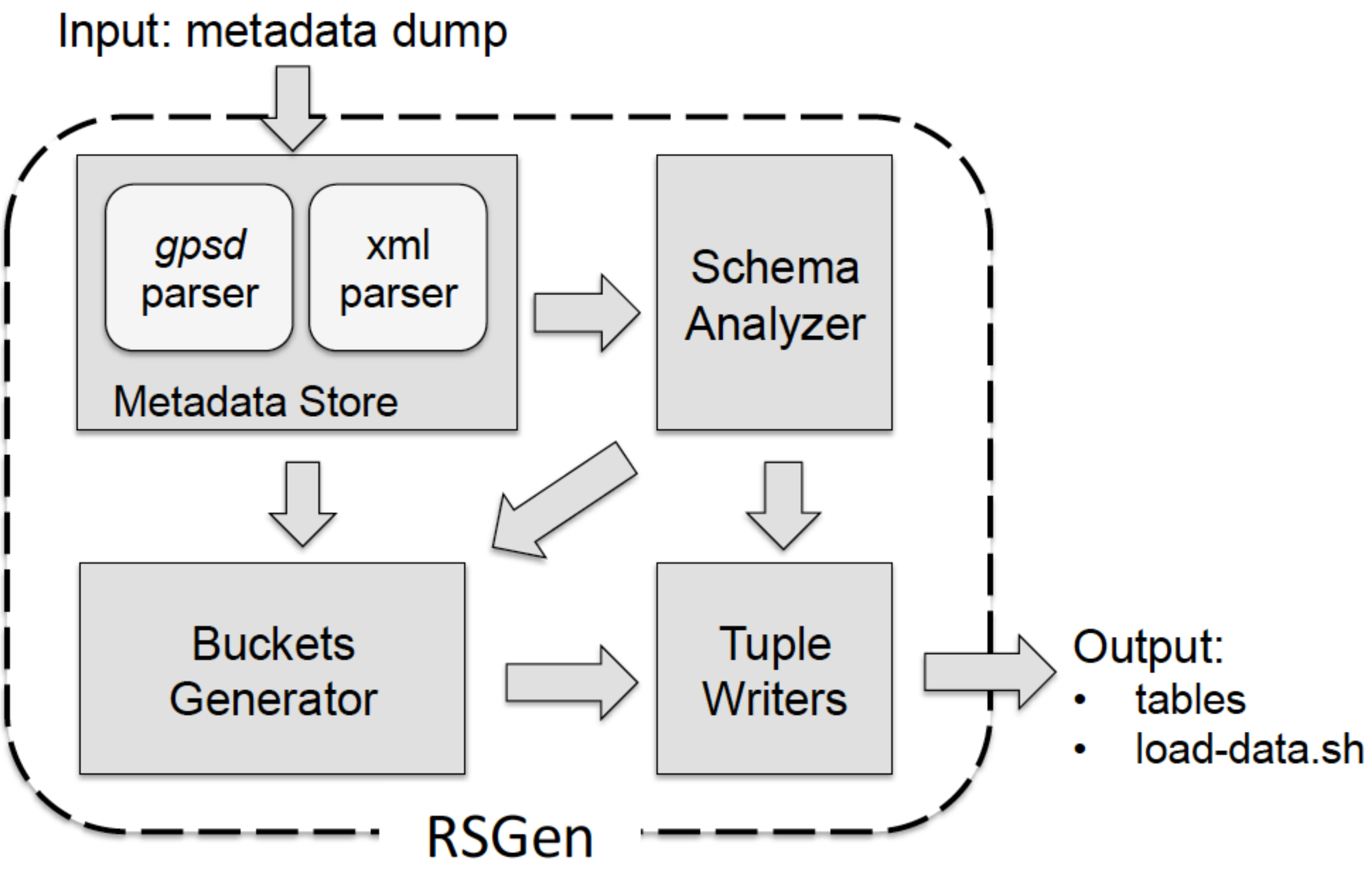
Pivotal

# Reversing Metadata

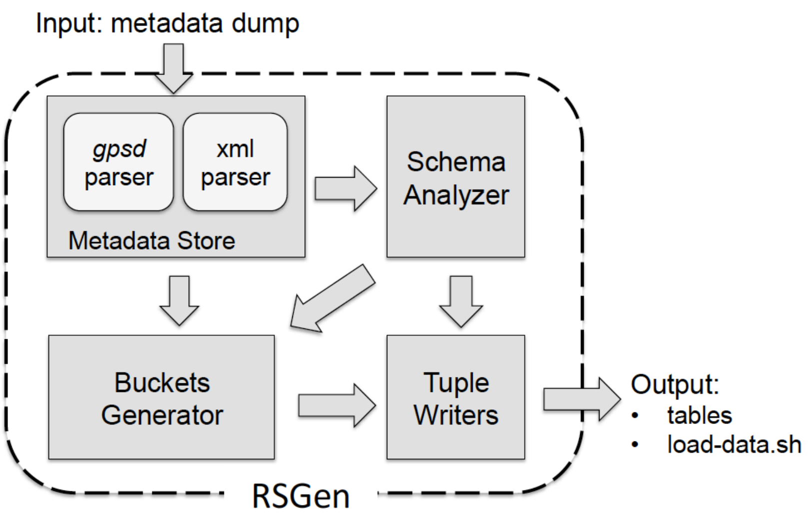
- Metadata: schemas, integrity constraints, statistics
  - Less sensitive, easier to anonymize
  - Much smaller in size
  - Retains significant amount of information
- Statistics in Greenplum Database
  - Collected by ANALYZE command
  - Only estimates (based on samples)
  - Column stats:
    - NULL fraction
    - Number of distinct values
    - Most common values (MCVs) and their frequencies (MCFs)
    - Equi-depth histogram

Pivotal

# Architecture



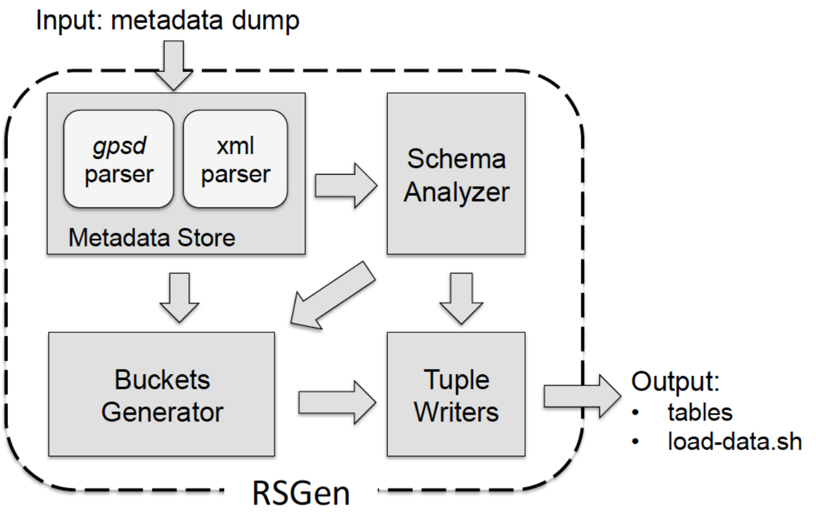
# Architecture




- Metadata Store
  - Parse and maintain metadata
  - System-independent
  - `gpsd` and xml parser
- Schema Analyzer
  - Interpret and maintain referential integrity among columns
  - Referential graph (DAG)
  - Topologically sort -> ordered lists of columns

Pivotal

# Architecture




- Bucket Generator
  - Simple but powerful and versatile
  - Central to massive parallelism
  - ‘Bucket alignment’

```
public class Bucket {
    Datum low;
    Datum high;
    long count;
    long nDistinct;}
```

- Tuple Writer
  - `Datum` type mapping (e.g. Date)
  - Column-oriented or row-oriented
    - Intra-table constraints
    - Parallel data loading

Pivotal

# Scalability and Extensibility

- Bucket model - two layers of parallelism
  - Tables generated in parallel after bucket alignment
  - Different slices within a table generated in parallel
  - Works for common scenarios, may be impossible for complex constraints
- Extensibility
  - Four modules relatively independent
  - Vertical extension: support improvement in statistical model
  - Horizontal extension: support other sources of input metadata
    - e.g. MaxDiff histogram

Pivotal

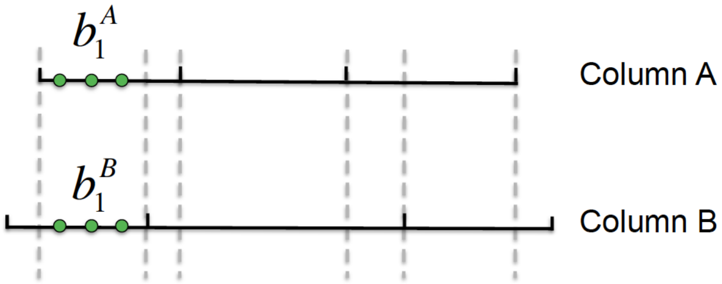
# Details – Bucket Generation

1. Process NULL fraction $nf$:
   - `low`: `null`, `high`: `null`, `count`: $N*nf$, `nDistinct`: 1
2. Process each MCV $v$ with frequency $f$:
   - `low`: $v$, `high`: $v$, `count`: $N*f$, `nDistinct`: 1
3. Process histogram:
   - direct mapping of histogram buckets
   - need to remove MCVs in the histogram
   - compute `nDistinct` and `count` for each bucket

Pivotal™

# Details – Handling integrity constraints

- Biggest challenge of scalability - generate referential columns in an independent manner
  - Bottleneck: reading the referred tuples from disk
  - [Rabl et al. 2011, Alexandrov et al. 2012]: Compute reference by a replaying the PRNG with certain seed and offset
  - Our solution: bucket alignment

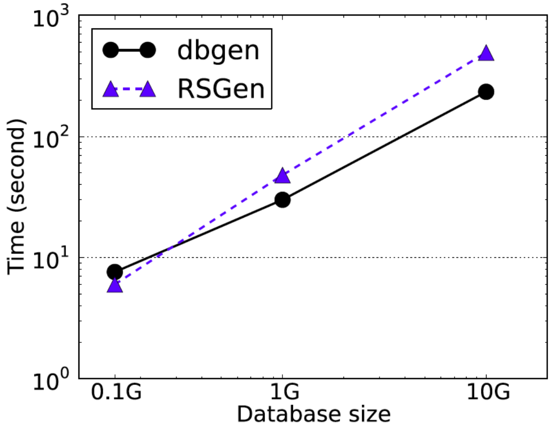


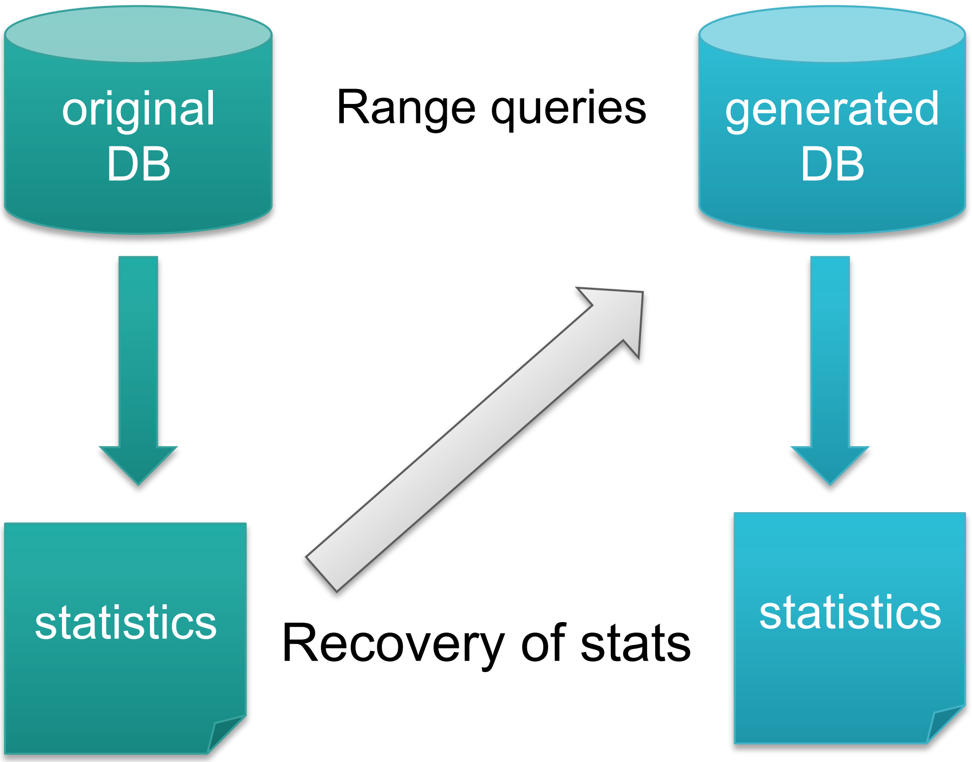
# Details – Handling integrity constraints


- e.g. column A refers to column B
  - Histogram bucket boundaries aligned ( 5 for A, 7 for B )
  - Bucket in the same 'position' having the same `nDistinct`
  - Values within an aligned bucket generated deterministically
    - An aligned bucket further divided into `nDistinct` intervals
    - A fix point from each interval will be generated
    - Same possible values from both columns

# Performance Evaluation

- Setup
  - Single node edition of Greenplum database 4.2.2
  - Test database: TPC-H with various scale factors
  - 1.8G Quad CPU, 8G memory
  - Implemented in Java
- Run time and scalability
  - RSGen (Java) vs dbgen (C)
  - Parallelism not fully exploited in this experiment yet




original DB
Range queries
generated DB
statistics
Recovery of stats
statistics

# Recovery of Statistics

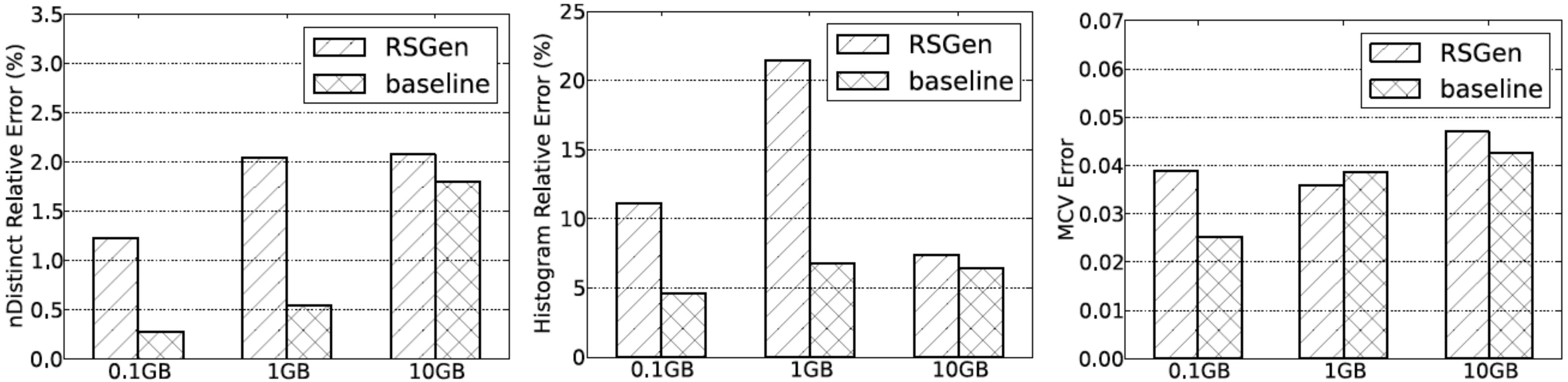
- Comparing the stats collected by `gpsd` from both databases
- Metrics
  - `nDistinct` relative error: average across all columns
  - Histogram error: average relative displacement of histogram boundaries
  - MCV error
    - Both MCVs and corresponding MCFs are considered
    - The case that multiple MCVs have the same frequency is considered too

$$
\begin{aligned}
&\sum\nolimits_{s\in\{MCV\cap MCV'\}}|f_s - f_{s'}| \\
&+\sum\nolimits_{s\in\{MCV\setminus MCV'\}, f_s>\min(MCF)} f_s \\
&+\sum\nolimits_{s\in\{MCV'\setminus MCV\}, f_s>\min(MCF')} f_s
\end{aligned}
$$

Pivotal

# Recovery of Statistics

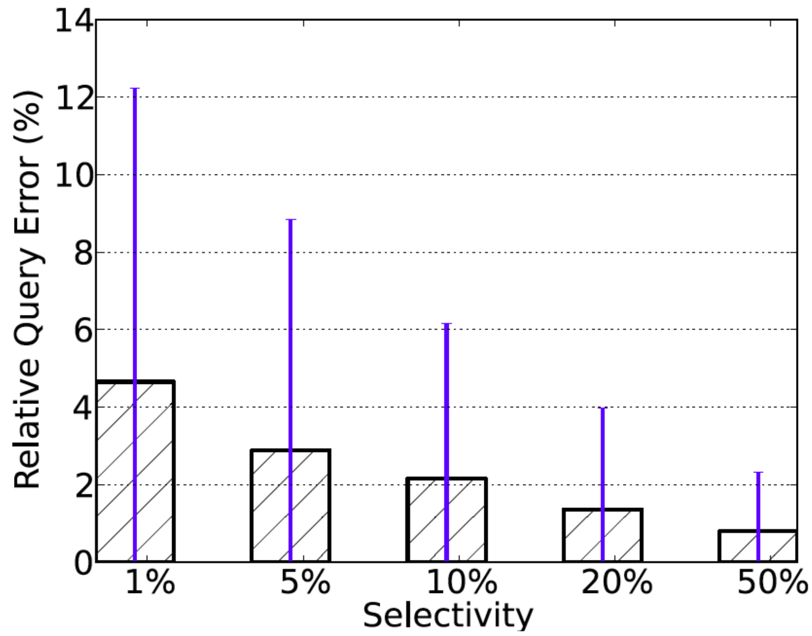
- Stats changes every time ANALYZE runs
- Baseline: achievable minimum error
- Observations




(a) Recovery of nDistinct (b) Recovery of histogram (c) Recovery of MCVs

# Range queries

- COUNT queries
  - A more pragmatic metric
  - Various selectivities
  - Average relative error less than 5%



# Work in progress

- Experiment on complex SQL queries
- Performance test on clusters
- Fine tune input metadata to capture different properties/ distributions not captured by the statistic model
  - Potential benefit for performance test on query optimizer

# Thank you