CODD: COnstructing Dataless Databases

Ankur Gupta
Database Systems Lab
Indian Institute of Science
Bangalore

†In archaic English, “cod” means “empty shell”
Testing Database Components

- Ever increasing data, evolving customer requirements, advancing systems
- Test Benchmarks evolve in diversity and complexity to cope with these demands
- Limitations:
  - Time and Space required to create and maintain large test datasets
  - DATA is the culprit
Can we do without DATA?

• Yes We Can! (with due apologies to Obama)
• To what extent?
  • Testing modules (whose inputs are solely metadata) such as
    • Query Plan Generators
    • Schema Advisors
    • System monitors

• Example Scenario: A query optimizer developer wishes to evaluate her module on a futuristic BigData setup featuring Yottabyte ($10^{24}$) sized relations.

• Can her wish be fulfilled?
CODD Metadata Processor

- CODD is a (free) graphical tool that attempts to alleviate the time and space constraints through construction of “dataless databases”

- Specifically, it facilitates
  - Automated Construction
  - Verification
  - Scaling
  - Retention
  - Porting of relational metadata configurations

- Operational on IBM DB2, Oracle, Microsoft SQLServer and HP NonStop SQL/MX

- Built in Java and contains about 50K Lines of Code
1) Metadata Generation

• Construct Mode

• Commercial engines provide techniques to manually update the catalog entries

• CODD leverages on them but adds value by
  • packaging them in largely vendor-neutral interface
  • adding functionalities such as validation, scaling, inter-engine transfer
Metadata Generation [Contd.]

- Comprises of statistics on:
  - Relational Tables (row cardinality, row length, no. of disk blocks)
  - Attribute Columns (column width, no. of distinct values, value distribution histograms)
  - Attribute Indexes (number of leaf blocks, clustering factor)
Metadata Generation [Contd.]

- CODD allows user to input/edit/upload all or any of these statistics
- Provides contextual on-line guidance for each input field
- Inputting/Modifying distribution statistics may be laborious and error prone

- Graphical Histogram
  - CODD provides graphical histogram editing interface to facilitate the modification of distribution statistics
Codd in Action!

Construct Mode.avi
Select `c_count`, `count(*)` as custdist
From (select `c_custkey`,
count(`o_orderkey`) as `c_count`
From `customer`
left outer join `orders` on `c_custkey` = `o_custkey`
and `o_totalprice` :varies
group by `c_custkey`
) as `c_orders`
group by `c_count`
order by `custdist` desc,
`c_count` desc

Query Template 13
2) Metadata Validation

Can user input arbitrary values?

• Need to ensure that input information is both
  • *Legal* – valid type and range
  • *Consistent* – compatible with other metadata values

• Validation Approach
  • Construct a *directed acyclic constraint graph* \( CG(V, E) \)
  • \( V \) represent set of single metadata entities, while \( E \) represents the set of statistical value dependencies
  • *Super Nodes*: used to represent collapsed chain of nodes for compactness
  • Run topological sort on \( CG \) to obtain \( CG_{linear} \)
  • CODD uses this linear ordering to guide the user
Metadata Validation [Contd.]

Sample Constraint Graph for a Commercial Database Engine

Legality Constraint

Statistical Dependency - Direction chosen as per hierarchy of abstraction

Dashed edges represent missing constraints

Signifies Order

Super Nodes

Relation level metadata
- Overflow (4)
- FPages (3)
- NPages (2)
- Card (1)
  - Integer type $\geq 0$ (or) -1

Index level metadata
- NLeafs (4)
- NLevels (17)
- Empty_Leafs (18)
- NumRIDs (15)
- Density (19)
- AvgColLenChar (7)
- High2Key (9)
- Low2Key (8)
- Quantile Value Distribution (11)
- Frequency Value Distribution (10)
Metadata Validation [Contd.]

Quantile Value Distribution

1. ColValue (1)
2. ValCount (2)
3. DistCount (3)
4. ColValue (4)
5. ValCount (5)
6. DistCount (6)
7. ColValue (7)
8. ValCount (8)
9. DistCount (9)
...
25. ColValue (25)
26. ValCount (26)
27. DistCount (27)
28. ColValue (28)
29. ValCount (29)
30. DistCount (30)

Frequency Value Distribution

1. ColValue (1)
2. ValCount (2)
3. DistCount (3)
4. ColValue (4)
5. ValCount (5)
6. DistCount (6)
7. ColValue (17)
8. ValCount (18)
9. DistCount (19)
...

Expanded Super-nodes for a Commercial Database Engine
3) Metadata Scaling

- Scaled version of the datasets are very commonly used in testing exercises.
- CODD supports two scaling models for metadata:
  - Space-based scaling (along the lines of TPC-H and TPC-DS)
  - Cost-based scaling - a novel approach
    - Aims to scale baseline metadata $M$ such that the optimizer’s estimated cost of executing query workload $Q$ on scaled version $M^\alpha$, is a specified multiple, $\alpha$, of the cost of executing it on $M$.

- **Initial Attempt:** Produce metadata so that cost of each individual query in $Q$ is scaled by $\alpha$.
  - May not always be feasible.
Metadata Scaling [Cost-Based]

• Optimization Problem

Produce an $M^\alpha$ such that the sum over $Q$ of the individual squared deviations from $\alpha$ in the cost scaling is minimized, subject to the constraint that the overall cost over $Q$ is scaled by $\alpha$

That is, given relations $\{R_1 \ldots R_h\} \in Q$, identify a relation cardinality-scaling vector $\{\alpha_1 \ldots \alpha_h\}$ such that

$$\sum_{q_i \in Q} \left[ \frac{c_{q_i}^s}{c_{q_i}^o} - \alpha \right]^2$$

is minimized subject to

$$\sum_{q_i \in Q} c_{q_i}^s = \alpha \cdot \sum_{q_i \in Q} c_{q_i}^o$$

• Complex mathematical relation between cost of query plans and scaling factors makes the above problem hard
Metadata Scaling [Cost-Based]

• Assumptions
  • Domains and distributions of attribute values remains same
  • Choice of query plan is retained
  • Simple polynomial expressions suffice to express relationships between operator costs and input sizes.

**Lemma:**

*If the key columns of relations are domain scaled and the primary key columns \((C_a, ..., C_n)\) of relation \(R_i\) are a combination of foreign key columns, which are referencing the relations \((R_a, ..., R_n)\), respectively, then the scaling factor \(\alpha_k\) of relation \(R_k\) is bounded by the product of \(\alpha_a, ..., \alpha_n\) where \(\alpha_a, ..., \alpha_n\) are the scaling factors of relations \((R_a, ..., R_n)\), respectively.*

• Example: PARTSUPP(PS_PARTKEY, PS_SUPPKEY)
Metadata Scaling [Cost-Based]

- **Algorithm for Cost Scaling**

  Input: Metadata Instance $M$, Query workload $Q$, Scaling factor $\alpha$
  Result: Scaled Metadata Instance $M^\alpha$

1. Determine $c_{qi}^O$, from the optimizer’s execution plan.
2. Determine $c_{qi}^S$, as an algebraic function of the scaling factors $[\alpha_1, \ldots, \alpha_h]$
3. Solve the optimization problem subject to two more constraints:
   
   for $k$ between 1 and $h$
   
   $0 \leq \alpha_k \leq \text{Lemma2 bound, if applicable}$
   
   $0 \leq \alpha_k \leq \infty, \text{otherwise}$

4. If a set of solutions $T$ is obtained in Step 3, pick a solution $t \in T$ that minimizes $\sum_{\alpha_k \in T} (\alpha - \alpha_k)^2$
5. Scale the input relations with the scaling factors obtained in Step 4 to get the required cost scaled metadata, $M^\alpha$
Example Scenario

• Setup: TPC-H 1 GB (baseline) and TPC-H 100TB (scaled)
  (both are metadata shells created using CODD)
• Machine: Laptop with 64GB Hard Disk
• Plan Diagram: Given a parameterized SQL query template that
defines a relational selectivity space, and a choice of database
engine, a plan diagram is a visual representation of the plan
choices made by the optimizer over this parameter space.

```
select n_name, o_year, sum(amount)
from (select n_name, o_orderdate, l_extendedprice
  from part, supplier, lineitem, partsupp, orders,
  where s_suppkey = l_suppkey and ps_suppkey = l_suppkey
  and ps_partkey = l_partkey and p_partkey = l_partkey and o_orderkey
  = l_orderkey and s_nationkey = n_nationkey and p_name like
  %green% and s_acctbal :varies and ps_supplycost :varies
  ) as all_nations
group by n_name, o_year
order by n_name, o_year desc
```
Example Scenario [Contd.]

No. of Plans increased from 32 to 77!

Significant change in geometries of optimal plan regions

Baseline Dataset (1GB)

Scaled Dataset (100TB)
Additional Features

• Inter-Engine Metadata Portability
  • Used for early assessments of impact of data migration

• Retain Mode
  • Used when initial metadata is required to be explicitly created from data
  • Allows reclamation of storage space by dropping some or all data
  • Requires careful handling of deletion triggers, key constraints and order of truncating relations
Applications

• Debugging legacy applications

• Identifying hidden constraints in database engine code:
  • For a popular commercial query optimizer, with the database size increasing in each iteration, we discovered that the cardinality estimation module “saturated”, when the input data size exceeded 10 exabytes
  • No mention of this threshold was found in the publicly available documentation of the system

• Doing comparative study of a query optimizer on different scale factors of a dataset

➤ Operational in Industry for testing of query optimizers
Future Directions

- CODD currently supports only meta-data based testing modules

What if we cannot do without DATA?

- Coupling with data generation frameworks like QAGen to facilitate execution module testing
- Adding support to modify the hardware configurations of the system, which will allow us to simulate enhanced system configurations
More Details

• For publication, software and documentation, please visit:
  http://dsl.serc.iisc.ernet.in/projects/CODD/index.html
THANK YOU!

Save Time. Save Space. Go Dataless.