# SQL Query Volume Performance Estimation Tool(DEMO)

Rekha Singhal
TCS Research
Mumbai, India.
rekha.singhal@tcs.com

Chetan Phalak
TCS Research
Mumbai, India.
chetan1.phalak@tcs.com

## ABSTRACT

Typically, applications are tested on small data size for both functional and non functional requirements. However, in production environment, the applications, having SQL queries, may experience performance violations due to increase in data volume. There is need to have tool which could test SQL query performance for large data sizes without elongating application testing phase. In this paper, we have presented a tool for estimating SQL query execution time for large data sizes without actually generating and loading the large volume of data. The model behind the working of the tool has been validated with TPC-H benchmarks and industry applications to predict within 10% average prediction error. The tool is built using underlying popular open source project CoDD with better project management and user interfaces.

## 1. INTRODUCTION

Most of the reporting and analytic applications constituting SQL queries may experience performance degradation in production environment. One of the prime reason is the increase in the data volume in production system whereas, the application was tested on small data size during its development. A naive approach to ensure performance assurance is to generate large data size, load it in the database and execute application's SQL queries to check the performance violations. However, this increases the cost to deployment by elongating the application testing phase and huge investment in building large size infrastructure in testing environment. There is a need to have tool which could estimate a SQL query execution time in testing phase for large data size without actually generating and loading the large volume of data. In this paper we have presented a tool which could estimate a SQL query (or set of SQL queries) execution time for large data sizes in relational databases without elongating testing cycle time significantly. The tool may be used by an application developer in testing environment to estimate a SQL query execution time in isolation for large data volume without actually generating it in the same environment. Currently, model is built assuming serial execution of SQL query execution plan, however, work has been done on extending it for parallel execution as well. The model behind the working of the tool has been validated for TPC-H benchmarks and industrial applications to predict within 10% average prediction error [4]. The output of model and tool for a given SQL query and projected data size is shown in Fig 2. The tool has provision for application project management to manage the testing of different application components for different data sizes. The tool has automated various processes of building and applying the model for SQL query performance estimation. The tool supports database volume extrapolation across different types of relational databases such as Postgres and Oracle. Our tool is an extension of popular open source project Constructing Dataless Database (CoDD) [1] as shown in Fig 1. The paper is organized as follows. Section 2 presents in brief the model used by the tool for estimating SQL execution time for large data sizes. Section 3 talks about the tool in detail including its features and implementation. Finally the paper is concluded in Section 4.

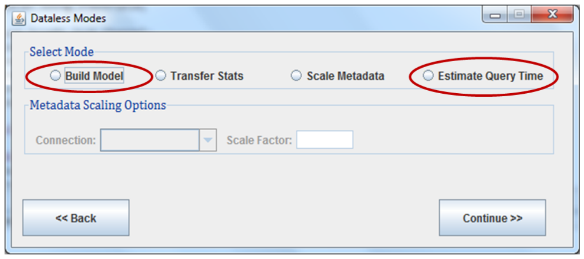


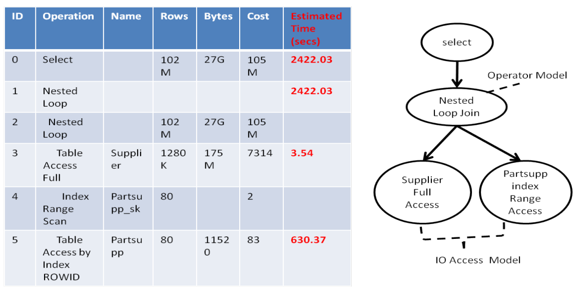
**Figure 1: Extension to CoDD for SQL Query Performance Estimation**

## 2. SQL QUERY VOLUME PERFORMANCE MODEL

The model for estimating a SQL execution time for large data volume has following steps.

- Define elementary steps in a SQL query execution plan as shown in Fig 2. Build models for estimating time taken by each elementary step, as function of the input(s) data size, in the testing environment- IO model is built as function of data size for estimating data access time using Full scan, Index scan, Primary Index and Secondary Index scan [3]. Operator Model

*ICPE'17 April 22-26, 2017, L'Aquila, Italy*
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4404-3/17/04.
DOI: http://dx.doi.org/10.1145/3030207.3053663



| ID | Operation | Name | Rows | Bytes | Cost | Estimated Time (secs) |
|---|---|---|---|---|---|---|
| 0 | Select | | 102 M | 27G | 105 M | 2422.03 |
| 1 | Nested Loop | | | | | 2422.03 |
| 2 | Nested Loop | | 102 M | 27G | 105 M | |
| 3 | Table Access Full | Supplier | 1280 K | 175 M | 7314 | 3.54 |
| 4 | Index Range Scan | Partsupp_sk | 80 | | 2 | |
| 5 | Table Access by Index ROWID | Partsupp | 80 | 11520 | 83 | 630.37 |

select
Operator Model
Nested Loop Join
Supplier Full Access
Partsupp index Range Access
IO Access Model

**Figure 2: SQL query Execution Plan with Estimated Execution Time and Elementary Steps**

is built as function of data sizes of its inputs for Aggregate, Hash Join, Sort, Sort Merge Join and Nested Loop join [4]. Synthetic SQLs are executed on small data size to collect IO access and SQL operator specific measurements in the underlying database.

- Build database volume emulator for the projected data size with given data growth patterns across relational tables [2, 1].

- For each SQL query- Get the SQL query execution plan from DB volume emulator.Map the SQL execution plan to set of the defined elementary steps. Estimate input(s) data size for each elementary step i.e. cardinality [4]. Predict the SQL query execution time as summation of estimated execution time of each of its elementary steps.

## 3. TOOL ARCHITECTURE

The tool is divided into two components GUI and Core engine as shown in Fig 3. User connects to GUI to provide inputs required for tool execution and receive output generated by tool. Core engine connects to databases to set and fetch required content. GUI consists of three subcomponents- 'Repository' which store environment access credentials, model's synthetic SQL queries and scripts for building model(s), 'Session handler' manages user's session with core engine and 'Output screen' where user gets output of the tool. Core engine is set of four engines. 'Model builder' generates small size database, build model and stores model parameters in 'Repository'. 'Replicator' creates content less replica of testing DB schema, which is called as emulated schema. 'Statistics Extrapolator' takes scale factor as input and extrapolates the emulated schema statistics by the scale factor. 'SQL response time estimator' estimates execution time of set of SQL queries received from the user and sends the output to GUI for user.

### 3.1 Features

**User Interaction**: The tool facilitates easy interface for interaction with the tool such as Add/remove/open projects workspace, Enter environment credentials, selective table(s) volume emulation and different ways of input/output
**Platform Independent**: The tool is fully developed in java and distributed in form of jar (java archive) file. All components of tool including GUI screens, four types of engines and database connectors are developed in java. Tool

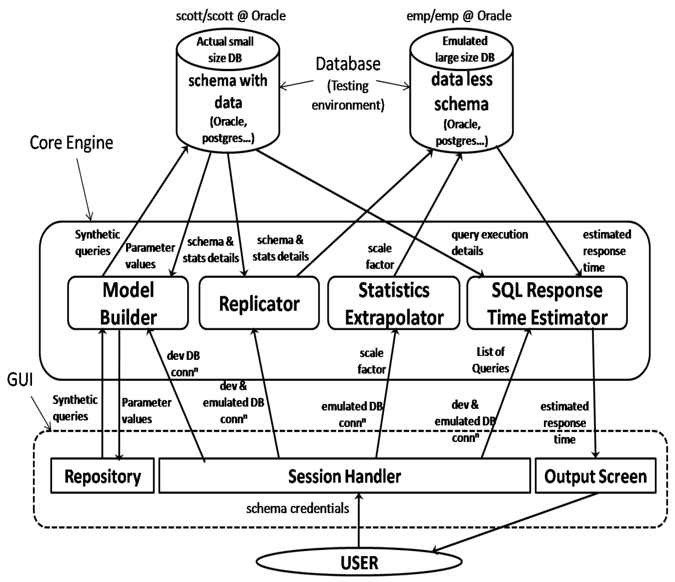


**Figure 3: SQL Query Volume Performance Estimation Tool Architecture**

package also contains other supporting files placed in repository. This repository contains only text files and jar files. All these facts make the tool platform independent.
**Project Management:** The provision of separate project workspace for different kind of databases or different schemas of one database helps user to easily distinguish and organize the various types of work in cohesive unit. Workspaces are very helpful in cases of complex scenarios when user need to maintain lots of emulated DBs with variant properties or configuration for different components of a large application.

## 4. CONCLUSIONS

In this paper, we have presented a tool architecture for estimating a SQL query (or set of SQL queries) execution time for large data volume in application testing environment. The tool is based on model which can emulate large data volume using database statistics and estimate SQL query execution time using IO access and Operator models as function of emulated data sizes [4]. The presented tool is an extensive extension of open source database volume emulator CoDD [1]. The tool supports project management, platform independence, database independence and GUI interface.

## 5. REFERENCES

[1] I. N. Rakshit S. Trivedi and J. R. Haritsa. Codd: Constructing dataless databases. In *In Proceedings of DBTest*, 2012.
[2] R. Singhal and M. Nambiar. Extrapolation of sql query elapsed response time at application development stage. In *In Proceedings of INDICON IEEE Proceedings*, 2012.
[3] R. Singhal and M. Nambiar. Measurement based model to study the effect of increase in data size on query response time. In *In Proceedings of Peformance and Capacity CMG*, 2013.
[4] R. Singhal and M. Nambiar. Predicting sql query execution time for large data volume. In *In Proceedings of IDEAS ACM Proceedings*, 2016.