[](http://tpc.org/default.asp)

TPC Express Big Bench

TPCx-BB

**Standard Specification**

Version 1.6.0

**September 2022**

Transaction Processing Performance Council (TPC)

www.tpc.org

info@tpc.org

© 2022 Transaction Processing Performance Council

All Rights Reserved

Legal Notice

The TPC reserves all right, title, and interest to this document and associated source code as provided under U.S. and international laws, including without limitation all patent and trademark rights therein.

Permission to copy without fee all or part of this document is granted provided that the TPC copyright notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the Transaction Processing Performance Council. To copy otherwise requires specific permission.

No Warranty

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE INFORMATION CONTAINED HEREIN IS PROVIDED “AS IS” AND WITH ALL FAULTS, AND THE AUTHORS AND DEVELOPERS OF THE WORK HEREBY DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY (IF ANY) IMPLIED WARRANTIES, DUTIES OR CONDITIONS OF MERCHANTABILITY, OF FITNESS FOR A PARTICULAR PURPOSE, OF ACCURACY OR COMPLETENESS OF RESPONSES, OF RESULTS, OF WORKMANLIKE EFFORT, OF LACK OF VIRUSES, AND OF LACK OF NEGLIGENCE. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, AND CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT WITH REGARD TO THE WORK.

IN NO EVENT WILL ANY AUTHOR OR DEVELOPER OF THE WORK BE LIABLE TO ANY OTHER PARTY FOR ANY DAMAGES, INCLUDING BUT NOT LIMITED TO THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA, OR ANY INCIDENTAL, CONSEQUENTIAL, DIRECT, INDIRECT, OR SPECIAL DAMAGES WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THIS OR ANY OTHER AGREEMENT RELATING TO THE WORK, WHETHER OR NOT SUCH AUTHOR OR DEVELOPER HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Trademarks

TPC Benchmark is a trademark of the Transaction Processing Performance Council.

Product names, logos, brands, and other trademarks featured or referred to within this Specification are the property of their respective trademark holders.

Acknowledgments

The TPC acknowledges the work and contributions of the TPC BigBench subcommittee member companies:

Jeffery Buell, Dave Rorke, Meikel Poess, Wayne Smith, John Poelman, Paul Cao, Matt Emmerton, Andy Bond, Da Qi Ren, Seetha Lakshmi, Tilmann Rabl, Nicholas Wakou, Yanpei Chen, Reza Taheri, Tariq Magdon-Ismail, Raghunath Nambiar, John Fowler, Bhaskar Gowda, Michael Brey, Jamie Reding, Doug Johnson, David Grimes, Chinmayi Narasimhadevara, Dileep Kumar, Francois Raab, Rodrigo Escobar, Jig Bhadaliya, William Benton.

TPC Membership

Find out more about the current members of the TPC in <http://www.tpc.org/TPC_Documents_Current_Versions/pdf/TPC_Membership.pdf>

Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | | Description |
| Februrary 19th  2016 | 1.0 | | TPCx-BB Sub Committee Voted Version 1.0 |
| Februrary 23rd  2016 | 1.0.1 | | TPCx-BB Sub Committee Editorial changes |
| May 16th  2016 | 1.1.0 | TPCx-BB Sub Committee Header Changes | |
| November 9th 2016 | 1.2.0 | TPCx-BB alignment with Pricing specification 2.0 | |
| June 6th 2019 | 1.3.0 | * Version/copyright update * Update TPC membership (convert to a link from a static image) * [0.2] Update URL link to kit * [4.1.1.6] Revised(corrected) number of rows in tables for different scale factors * [Appendix B.4] Refined queries’ description-to-implementation match. * [Appendix C] Corrected Q19 details to correct a date that was not parsed correctly in all implementations * [2.3.2.1/6.2/9.4.1.17] Clarify the intersection of “external to SUT” data generation in 2.3.2.1 and pricing wording in 6.2 with a linkback in auditor checklist. | |
| August 27th 2019 | 1.3.1 | * Version/date update * [0.3] typo fix * [0.5] remove misplaced wording regarding PDGF * [2.1.4.4] Update config file names to match kit and add comment indicating alternate ways to configure * [2.1.5.1] Update config file names to match kit * [2.3.1.1] typo fix * [2.4.1.1] Require new wrapper script to run benchmark sequence * [3.1.1] typo fix * [3.1.3.3] typo fix * [3.1.3.3] discuss how new wrapper collects redundancy info for Hive * [8.3.1.3] typo fix * [9.4.1.3] typo fix * [Appendix H] Update sample output for Hive redundancy check | |
| August 21st 2020 | 1.4.0 | * [0.2,/2.1.1/ 2.1.2] Equate users guide to README.md * [0.5] Add definitions for ADS, Data Node, EADS, High Availability System, IADS, Master Node, Name Node, Undo/Redo log. Update definitions of LCS, Metadata * [3.1.3.2] Add reference to Metadata * [3.1.3.3] Rework to specify requirements for non-distributed-filesystem solutions. Add a bit more detail regarding erasure coding and 3-way replication requirements. * [8.8.2.4/Appendix A] Add redundancy detail to Executive Summary (to highlight replication or erasure coding) * [9.3.2] Change PPB term to match TPC policies * [Appendix B] Change Q5 description to match 90/10 random split of data and fix typo * [Appendix C] Replace with current version from kit (includes Q5 shift from ‘Books’ to ‘Movies & TV’ * [Appendix D] Replace with current version from kit | |
| Feburary 5, 2021 | 1.5.0 | * Version/date update * All other changes are to kit to add support for CDP 7.1 and to remove previously deprecated support for CDH 5.x and HDP 2.x | |
| September 9th, 2021 | 1.5.1 | * Version/date update. * Update to sub-committee members in aknowledgements. * All other changes are to kit to restore -verboseTableName parameter for Maxcompute. | |
| February 8th, 2022 | 1.5.2 | * Version/date update * Update to sub-committee members in aknowledgements. | |
| July 22nd, 2022 | 1.6.0 | * Version/date update * [3.1.3.3] Fix typo (medum/medium). Adds requirements on HAS redundancy when continued execution cannot be guaranteed due to a medium failure. * [3.1.3.3.b] Grammar improvement on requirements for distributed file systems other than HDFS. Emphasizes that solutions using distributed file systems other than HDFS must still show equivalence to HDFS with at least 3-way replication. * [4.1.3.6] Updates version of OpenNLP to 1.9.3 | |

Typographic Conventions

The following typographic conventions are used in this specification:

|  |  |  |
| --- | --- | --- |
| Convention | | Description |
| Bold | | Bold type is used to highlight terms that are defined in this document |
| *Italics* | | Italics type is used to highlight a variable that indicates some quantity whose value can be assigned in one place and referenced in many other places. |
| UPPERCASE | | Uppercase letters names such as tables and column names. In addition, most acronyms are in uppercase. |
|  | | |

Table of Contents

[Clause 0 -- Preamble 11](#_Toc63428150)

[0.1 Introduction 11](#_Toc63428151)

[0.1.1 Restrictions and Limitations 11](#_Toc63428152)

[0.2 TPCx-BB Kit and Licensing 11](#_Toc63428153)

[0.3 General Implementation Guidelines 12](#_Toc63428154)

[0.3.1 Benchmark Specials 12](#_Toc63428155)

[0.3.2 Benchmark Special Characteristics 12](#_Toc63428156)

[0.4 General Measurement Guidelines 13](#_Toc63428157)

[0.5 Definitions 13](#_Toc63428158)

[Clause 1 -- Overview 20](#_Toc63428159)

[1.1 Overview of Data Model 20](#_Toc63428160)

[1.1.1 Structured Data 20](#_Toc63428161)

[1.1.2 Semi-structured and Unstructured Data 20](#_Toc63428162)

[1.1.3 Queries 21](#_Toc63428163)

[Clause 2 -- WORKLOAD AND EXECUTION 22](#_Toc63428164)

[2.1 Benchmark Kit 22](#_Toc63428165)

[2.1.1 Kit Contents 22](#_Toc63428166)

[2.1.2 Kit Usage 22](#_Toc63428167)

[2.1.3 Kit Run report 22](#_Toc63428168)

[2.1.4 Kit Parameter settings 22](#_Toc63428169)

[2.1.5 Test Sponsor Kit Modifications 23](#_Toc63428170)

[2.2 Benchmark Kit Modifications 23](#_Toc63428171)

[2.2.1 Simple Review of Kit Modifications 24](#_Toc63428172)

[2.2.2 Formal Review of Kit Modifications 24](#_Toc63428173)

[2.2.3 Kit Validation 25](#_Toc63428174)

[2.2.4 Classification of Major, Minor and Third Tier Kit Modifications 25](#_Toc63428175)

[2.3 Benchmark Run 26](#_Toc63428176)

[2.3.2 Load Test 27](#_Toc63428177)

[2.3.3 Power Test 28](#_Toc63428178)

[2.3.4 Throughput Test 28](#_Toc63428179)

[2.4 Benchmark Execution 28](#_Toc63428180)

[2.5 Configuration and Tuning 29](#_Toc63428181)

[Clause 3 – System Under Test 30](#_Toc63428182)

[3.1 Logical Breakdown of System Under Test 30](#_Toc63428183)

[3.1.1 System Under Test 30](#_Toc63428184)

[3.1.2 Commercially Available Products 31](#_Toc63428185)

[3.1.3 Data Redundancy Requirement 31](#_Toc63428186)

[Clause 4 -- SCALE FACTORS and Result validation 34](#_Toc63428187)

[4.1 Scale Factor 34](#_Toc63428188)

[4.1.2 Result Validation 35](#_Toc63428189)

[4.1.3 Output data for Validation test. 36](#_Toc63428190)

[Clause 5 Metrics 38](#_Toc63428191)

[5.1 TPCx-BB Primary Metrics 38](#_Toc63428192)

[5.2 Performance Metric (BBQpm@SF) 38](#_Toc63428193)

[5.3 Price Performance Metric ($/BBQpm@SF) 39](#_Toc63428194)

[5.4 System Availability Date 39](#_Toc63428195)

[5.5 Fair Metric Comparison 39](#_Toc63428196)

[5.6 Secondary Metrics 39](#_Toc63428197)

[Clause 6 Pricing 41](#_Toc63428198)

[6.1 Introduction 41](#_Toc63428199)

[6.1.1 Pricing Methodology 41](#_Toc63428200)

[6.2 Priced Configuration 41](#_Toc63428201)

[6.3 Additional Operational Components 41](#_Toc63428202)

[6.4 Allowable Substitutions 42](#_Toc63428203)

[Clause 7 – ENERGY 43](#_Toc63428204)

[Clause 8 -- Full Disclosure Report 44](#_Toc63428205)

[8.1 Full Disclosure Report Requirements 44](#_Toc63428206)

[8.2 Format Guidelines 44](#_Toc63428207)

[8.3 General Items 44](#_Toc63428208)

[8.4 Software Components and Dataset Distribution 46](#_Toc63428209)

[8.5 Workload Related Items 48](#_Toc63428210)

[8.6 SUT Related Items 48](#_Toc63428211)

[8.7 Metrics and Scale Factors 49](#_Toc63428212)

[8.8 Audit Related Items 49](#_Toc63428213)

[8.8.2 Implementation Overview 50](#_Toc63428214)

[8.8.3 Pricing Spreadsheet 51](#_Toc63428215)

[8.8.4 Numerical Quantities Summary 52](#_Toc63428216)

[8.8.5 TPCx-BB Run Report 53](#_Toc63428217)

[8.9 Availability of the Full Disclosure Report 53](#_Toc63428218)

[8.10 Revisions to the Full Disclosure Report 53](#_Toc63428219)

[Clause 9 – Auditing 54](#_Toc63428220)

[9.1 TPC Pricing 54](#_Toc63428221)

[9.2 Optional TPC-Energy Results 54](#_Toc63428222)

[9.3 General Rules 54](#_Toc63428223)

[9.3.1 Independent Audit 54](#_Toc63428224)

[9.3.2 Pre-Publication Board 54](#_Toc63428225)

[9.3.3 Results Based on Existing TPCx-BB Results 54](#_Toc63428226)

[9.4 Audit Checklist 55](#_Toc63428227)

[Appendix A. Sample Executive Summary 57](#_Toc63428228)

[Appendix B. Logical Database Design 61](#_Toc63428229)

[B.1 Table Columns Used by Queries 61](#_Toc63428230)

[B.1.1 Variables 66](#_Toc63428231)

[B.2 Table Data Generation Rules 71](#_Toc63428232)

[B.2.1 Data Generation 91](#_Toc63428233)

[B.3 Query Overview 91](#_Toc63428234)

[B.3.1 Query types 91](#_Toc63428235)

[B.3.2 Query Grouping 92](#_Toc63428236)

[B.4 Query Descriptions 92](#_Toc63428237)

[B.4.1 Schema 96](#_Toc63428238)

[B.4.2 Weighted lists 96](#_Toc63428239)

[Appendix C. -- Query Parameters 97](#_Toc63428240)

[Appendix D. – Benchmark Parameters 101](#_Toc63428241)

[Appendix E. – Global Framework Parameters 104](#_Toc63428242)

[Appendix F. – Local Settings Parameters 108](#_Toc63428243)

[Appendix G. – SUT Hardware and Software 109](#_Toc63428244)

[Appendix H. – Data Redundancy Report 117](#_Toc63428245)

[Appendix I. – Custom Load Script 118](#_Toc63428246)

[Appendix J. – Throughput Test Stream Placement 136](#_Toc63428247)

Table of Figures

[Figure 1 TPCx-BB Data Model 20](#_Toc63428248)

[Figure 2 System under Test 31](#_Toc63428249)

[Figure 3 Sample Configuration Diagram 46](#_Toc63428250)

Table of Tables

[Table 1- Scale Factors 34](#_Toc63428251)

[Table 2-1 Dataset table sizes 35](#_Toc63428252)

[Table 3 Example Layout Description 47](#_Toc63428253)

[Table 4 Sponsor and System Identification 50](#_Toc63428254)

[Table 5 Benchmark Results 50](#_Toc63428255)

[Table 6 System Configuration Information 51](#_Toc63428256)

[Table 7 Storage and Memory Ratios 51](#_Toc63428257)

[Table 8 Measurement Results for Performance Run 52](#_Toc63428258)

1. -- Preamble
   1. Introduction

Big data analytics is a growing field of research and business. The significant decrease in the overall cost of hardware, the emergence of Open Source based analytics frameworks, along with the greater depth of data mining capabilities allows new types of data sources to be correlated with traditional data sources. For example, online retailers used to record only successful transactions on their website, whereas modern systems are capable of recording every interaction. The former allowed for simple shopping basket analysis techniques, while the current level of detail in monitoring makes detailed user modeling possible. The growing demands on data management systems and the new forms of analysis have led to the development of a new type of **Big Data Analytics Systems** (**BDAS**).

Similar to the advent of **Database Management Systems**, there is a vastly growing ecosystem of diverse approaches to enabling Big Data Analytics Systems. This leads to a dilemma for customers of **BDAS**, as there are no realistic and proven measures to compare different **BDAS** solutions. To address this, TPC has developed TPCx-BB (BigBench), which is an express benchmark for comparing **BDAS** solutions. The TPCx-BB Benchmark was developed to cover essential functional and business aspects of big data use cases. The benchmark allows for an objective measurement of **BDAS** System under Test, and provides the industry with verifiable performance, price/performance, and availability metrics.

* + 1. Restrictions and Limitations

The extent to which a customer can achieve the **Result**s reported by a vendor is highly dependent on how closely the TPCx-BB measurements and configuration approximates the customer application. The relative performance of systems derived from these benchmarks does not necessarily hold for other workloads or environments. Extrapolations to any other environments are not recommended.

Benchmark **Result**s are highly dependent upon workload, specific application requirements, systems design, and implementation. Relative system performance and environments will vary because of these and other factors. Therefore, TPCx-BB **Results** should not be used as a substitute for specific customer application benchmarking when critical capacity planning and/or product evaluation decisions are considered.

**Test Sponsors** are allowed various possible implementation designs, insofar as they comply with the model described and illustrated in this specification, TPC Energy and Pricing specifications. A [**Full Disclosure Report (FDR)**](#full_disclosure_report) of the implementation details, as specified in Clause 8, must be made available along with the reported TPCx-BB metrics.

While separated from the main text for readability, comments are a part of the standard and must be enforced.

* 1. TPCx-BB Kit and Licensing

The TPCx-BB kit is available from the TPC website (see www.tpc.org/tpcx-bb/ for more information). Users must sign-up and agree to the TPCx-BB End User Licensing Agreement (EULA) to download the kit. All related work (such as collaterals, papers, derivatives) must acknowledge the TPC and include the TPCx-BB copyright. The TPCx-BB kit includes: TPCx-BB Specification document (this document), TPCx-BB Users Guide (README.md) documentation, shell scripts to set up the benchmark environment, Java code to execute the benchmark workload, Data Generator, **Query** files, and Benchmark Driver.

* 1. General Implementation Guidelines

The purpose of TPC benchmarks is to provide relevant, objective performance data to industry users. To achieve that purpose, TPC Benchmark Specifications require that benchmark tests be implemented with systems, products, technologies, and pricing that:

* are generally available to users
* are relevant to the market segment that the individual TPC benchmark models, or represents for example, TPCx-BB models and represents a Big Data Analytics System such as Hadoop ecosystem or Hadoop file system API compatible systems.
  + 1. Benchmark Specials

The use of new systems, products, technologies (hardware or software) and pricing is encouraged so long as they meet the requirements above. Specifically prohibited are benchmark systems, products, technologies, pricing (hereafter referred to as "implementations") whose primary purpose is optimization of TPC Benchmark **Results** without any corresponding applicability to real-world applications and environments. The intent is to disallow "**Benchmark Special**” implementations that improve benchmark results but not real-world performance, pricing, or energy consumption.

The following characteristics should be used as a guide to judge whether a particular implementation is a **Benchmark Special**. It is not required that each point below be met, but that the cumulative weight of the evidence be considered to identify an unacceptable implementation. Absolute certainty or certainty beyond a reasonable doubt is not required to make a judgment on this complex issue. The question that must be answered is this: based on the available evidence, does the clear preponderance (the greater share or weight) of evidence indicate that this implementation is a **Benchmark Special**?

* + 1. Benchmark Special Characteristics

The following characteristics should be used to judge whether a particular implementation is a **Benchmark Special**:

* Is the implementation generally available, documented, and supported?
* Does the implementation have significant restrictions on its use or applicability that limits its use beyond TPC benchmarks?
* Is the implementation or part of the implementation poorly integrated into the larger product?
* Does the implementation take special advantage of the limited nature of TPC benchmarks (e.g., limited duration, use of virtualized capabilities not found in the **Commercially Available Product**) in a manner that would not be generally applicable to the environment the benchmark represents?
* Is the use of the implementation discouraged by the vendor? (This includes failing to promote the implementation in a manner similar to other products and technologies.)
* Does the implementation require uncommon sophistication on the part of the end-user, datacenter facility manager, programmer, or system administrator?
* Does the implementation use knowledge of the variability of the possible components to enhance the **Result** in such a way as to be significantly different from what a typical customer would experience?
* Is the implementation being used (including beta) or purchased by end-users in the market area the benchmark represents? How many? Multiple sites? If the implementation is not currently being used by end-users, is there any evidence to indicate that it will be used by a significant number of users?
  1. General Measurement Guidelines

TPCx-BB **Results** are expected to be accurate representations of system performance. Therefore, there are certain guidelines that are expected to be followed when measuring those **Results**. The approach or methodology to be used in the measurements are either explicitly described in the specification or implemented by the TPCx-BB Kit (Clause 2.1). When not described in the specification, the methodologies and approaches used must meet the following requirements:

* The approach is an accepted engineering practice or standard.
* The approach does not enhance the **Result**s.
* The equipment used in measuring **Result**s must conform to the requirements in Clause 3.
* Fidelity and candor are maintained in reporting any anomalies in the **Result**s, even if not specified in the benchmark requirements.

The use of new methodologies and approaches is encouraged so long as they meet the requirements above.

* 1. Definitions

A \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Auxillary Data Structures (ADS)

Other than the base table data structures, any database structure that contains a copy of, reference to, or data computed from base table data is defined as an auxiliary data structures (ADS). The data in the ADS is materialized from the base table data; references are a form of materialization. There is an essential distinction between base table data contained in a base table data structure and data contained in auxiliary data structures. Because auxiliary data structures contain copies of, references to, or data computed from base table data, deleting data from an auxiliary data structure does not result in the loss of base table data in that it is still contained in the base table data structure. In contrast, deleting data from a base table data structure (in the absence of copies in any auxiliary data structures) does result in the loss of base table data.

There are two types of auxiliary data structures: Implicit and explicit. An explicit auxiliary data structure (EADS) is created as a consequence of a directive (e.g. DDL, session options, global configuration parameters). These directives are called EADS Directives. Any ADS which is not an EADS is by definition an Implict ADS(IADS).

Comment: In contrast to an IADS, an EADS would not have been created without the directive

Attestation Letter

**TPC-Certified Auditor**’s opinion regarding the compliance of a **Result** must be consigned in an Attestation Letter delivered directly to the **Test Sponsor**.

Availability Date

The **Availability Date** is the System **Availability Date** defined in the TPC Pricing Specification.

B \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Benchmark Special

The **Benchmark Special** is defined as any aspect of the benchmark implementation with the primary purpose of the optimization of TPC Benchmark **Results** without any corresponding applicability to real-world applications and environments.

BDAS

A **Big Data Analytics System** (BDAS) is a collection of commercially available software used to implement Big Data Analytics.

C \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Commercially Available Product

**Commercially Available Product** is defined in TPC Pricing Specification.

D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Data Redundancy

The ability to have no permanent data loss after the permanent irrecoverable failure of any single Durable Medium containing tables, input data, output data, or metadata.

Data Generation

The process of using **PDGF** to create the data in a format suitable for presentation to the load facility.

Data Node

***Data Nodes*** *store data in a Hadoop cluster and is the name of the daemon that manages the data. File data is replicated on multiple* ***Data Nodes*** *for reliability and so that localized computation can be executed near the data[[1]](#footnote-1).*

E \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Explicit Auxiliary Data Structure (EADS)

See Auxillary Data Structure (ADS).

Executive Summary

Defined by the TPC Policies, an **Executive Summary** is a two to four page summary of the Result.

F \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Full Disclosure Report (FDR)

The **Full Disclosure Report** is a set of files that documents how a benchmark Result was implemented and executed in sufficient detail so that the Result can be reproduced given the appropriate hardware and software products.

Framework

A Framework is a collection of software from **BDAS**, including API’s, distributed computing engines and libraries used to run TPCx-BB.

G \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

H \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HDFS

**HDFS** (Hadoop Distributed File System)is a file system that provides scalable and reliable data storage, and it was designed to span large clusters of commodity servers.

High Availability System

*Computing environments configured to provide nearly full-time availability are known as* ***High Availability Systems****. Such systems typically have redundant hardware and software that makes the system available despite failures. Well-designed high availability systems avoid having single points-of-failure. Any hardware or software component that can fail has a redundant component of the same type*[[2]](#footnote-2).

I \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Implicit Auxiliary Data Structure (IADS)

See Auxillary Data Structure (ADS).

J \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

JBOD

JBOD (Just a Bunch of Disks) refers to a collection of hard disks that have not been configured to act as a redundant array of independent disks (RAID) array.

K \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

L \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**LCS** (Licensed Compute Services)

**Licensed Compute Service (LCS)** is defined in TPC Pricing Specification: *Publicly offered processing, storage, network, and software services that are hosted on remote computer servers accessed via a Wide Area Network (e.g. the Internet). A Customer pays a license fee to the Licensed Compute Services vendor for the use of the processing, storage, network, and software services. The Licensed Compute Services are not located or installed on a Customer's premises.*

M \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Metastore/Metadata

Descriptive information about the database including names and definitions of tables, indexes, and other schema objects. Various terms commonly used to refer collectively to the **Metadata** include **Metastore**, information schema, data dictionary, or system catalog. **Metadata** also includes additional information managed by the **BDAS** and stored in the database to define, manage and use other database objects, e.g. users, connections, etc.

**Master Node**

**Master Node(s)** provide a variety of storage and processing coordination services for a cluster. These are conceptually distinct from **Data Nodes** but sometimes share physical hardware. Where necessary for correct execution of the benchmark, data in the **Master Node** services is considered **Metadata** unless it is data for a co-located **Data Node** service in which case it is considered table data/**EADS**. Some **Master Node** services can be configured with sufficient instances as part of a **High Availability System** while others require other approaches to protecting against loss of service or data loss. Examples of **Master Node** services include **Name Nodes**, Checkpoint Nodes, Journal Nodes, Resource Manager, Job Tracker, HMaster, Zookeeper[[3]](#footnote-3).

N \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Name Node**

**A Name Node** is a particular class of **Master Node** service. ***Name Nodes*** *maintain the namespace tree for* ***HDFS*** *and a mapping of file blocks to* ***Data Nodes*** *where the data is stored. A simple* ***HDFS*** *cluster can have only one primary* ***Name Node****, supported by a secondary* ***Name Node****[[4]](#footnote-4)*

O \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Operating System/OS

The term Operating System refers to a commercially available program that, after being initially loaded into the computer by a boot program, manages all the other programs in a computer, or in a **VM**. The Operating System provides a software platform on top of which all other programs run. Without the Operating System and the core services that it provides no other programs can run and the computer would be non-functional. Other programs make use of the Operating System by making requests for services through a defined application program interface (API). All major computer platforms require an Operating System. The functions and services supplied by an Operating System include but are not limited to the following:

* manages a dedicated set of processor and memory resources
* maintains and manages a file system
* loads applications into memory
* ensures that the resources allocated to one application are not used by another application in an unauthorized manner
* determines which applications should run in what order, and how much time should be allowed to run the application before giving another application a turn to use the systems resources
* manages the sharing of internal memory among multiple applications
* handles input and output to and from attached hardware devices such as hard disks, network interface cards, addon cards and other hardware devices.

Some examples of Operating Systems are listed below:

* Windows
* Unix (Solaris, AIX)
* Linux (Red Hat, SUSE)
* Mac OS

P \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Performance Metric

The reported throughput as expressed in BigBench **Queries** per minute.

Performance Run

The **Performance Run** is defined as the run with the lower TPCx-BB **Performance Metric** of the two TPCx-BB test runs**.**

Priced Configuration

The **Priced Configuration** consists of components defined in the TPCx-BB Benchmark Standard including all hardware, software and maintenance.

Price/Performance Metric

The **Price/Performance Metric** is the total price of the Priced Configuration divided by the TPCx-BB **Performance Metric**.

PGDF

The **PDGF** (**Parallel Data Generator Framework)** is part of TPCx-BB kit used to generate **Test Dataset**.

Q \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Query/ies

A Query is an implementation of one or more **Use Cases** comprised in the TPCx-BB.

R \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Repeatability Run

Of the two TPCx-BB test runs, the **Repeatability Run** is defined as the run with the higher TPCx-BB **Performance Metric.**

Report

The **Report** is an Adobe Acrobat PDF file in the **FDR**. The contents of the Report are defined in Clause 8.

Reported

The term **Reported** an item that is part of the **FDR**.

Result

A performance test, documented by a **FDR** and **Executive Summary** submitted to the TPC, claiming to meet the requirements of the TPCx-BB Benchmark Standard.

S \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Software Version

A **Software Version** uniquely identifies a software product, its release level, update level, and/or patch level. It is typically a string of alphanumeric characters that allows the software manufacturer to uniquely identify the software.

Substitution

Substitution is the use of components in the **Priced Configuration** which are different than those used in the measured configuration.

Supporting Files

Supporting Files refers to the contents of the **Supporting Files** folder in the **FDR**. The contents of this folder, consisting of various source files, scripts, and listing files, are defined in Clause 8.

System Under Test (SUT)

System Under Test (**SUT**) – is defined to be the sum of the components utilized in running a benchmark as specified in Clause 3.

T \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Test Sponsor

The Test Sponsor is the company officially submitting the **Result** with the **FDR** and will be charged the filing fee. Although multiple companies may sponsor a **Result** together, for the purposes of the TPC’s processes the Test Sponsor must be a single company. A Test Sponsor need not be a TPC member. The Test Sponsor is responsible for maintaining the **FDR** with any necessary updates or corrections. The Test Sponsor is also the name used to identify the **Result**.

Test Dataset

The **Test Dataset** is the data generated by **PDGF** for the defined scale factor used in the test.

Test Database

The **Test Database** is the database used to execute the database Load test, Power test and Throughput test.

TPC-Certified Auditor (Auditor)

The term TPC-Certified Auditor is used to indicate that the TPC has reviewed the qualification of the **Auditor** and has certified his/her ability to verify that benchmark Results are in compliance with a specification. (Additional details regarding the **Auditor** certification process and the audit process can be found in Section 9 of the TPC Policies document.)

U \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Undo/Redo Log**

**Undo/Redo Log**: records all changes made in data files. The **Undo/Redo Log** makes it possible to replay all the actions executed by the **BDAS**. If something happens to one of the data files, a backed up data file can be restored and the **Undo/Redo Log** that was written since the backup can be played and applied which brings the data file to the state it had before it became unavailable. Not all BDAS environments utilize an **Undo/Redo Log** to accommodate recovery.

Use Case

A Use Case defines a single problem solved by the Big Data Analytics System. It is Framework and syntax agnostic and can be implemented in many ways. In theTPCx-BB kit all **Use Cases** are implemented in the form of **Queries**.

V \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

W \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

X \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Y \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Z \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. -- Overview
   1. Overview of Data Model

TPCx-BB is an application benchmark for Big Data based on paper “BigBench: Towards an Industry Standard Benchmark for Big Data Analytics”\*. This choice highly sped up the development of TPCx-BB and made it possible to start from a solid and proven foundation. A high-level overview of the data model is presented in Figure 1.

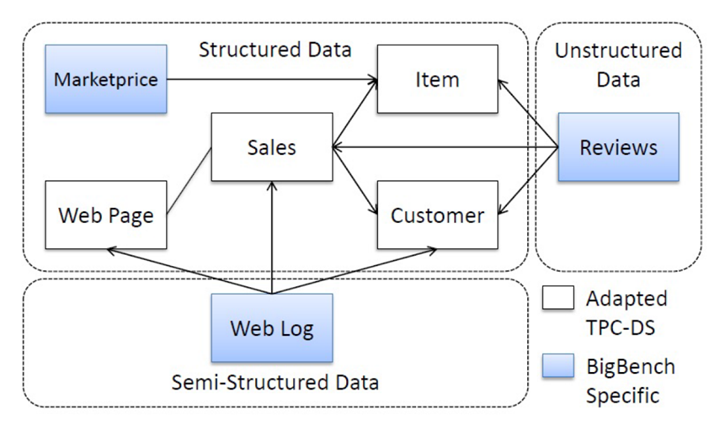


Figure 1 TPCx-BB Data Model

* + 1. Structured Data

TPCx-BB is designed with a multiple-snowflake schema inspired by TPC-DS using a retail model consisting of five fact tables, representing three sales channels, store sales, catalog sales, and online sales, each with a sales and a returns fact table. As shown in Figure 1, big data specific dimensions were added. The Marketprice is a traditional relational table storing competitors' prices.

Figure 1 only shows a subset of the TPCx-BB Data Model. For example, Figure 1 does not include all fact tables

* + 1. Semi-structured and Unstructured Data

Structured, semi-structured and unstructured data are very different. Structured data accounts for only 20% of the data available. It is clean, analytical and usually stored in databases. Semi structured data is a form of structured data that does not conform to formal structure of data models. The idea of utilizing unstructured data for analysis has in the past been far too expensive for most companies to consider. Thanks to technologies such as Hadoop, unstructured data analysis is becoming more common in the business world. Business owners may be wondering if the use of unstructured data could give them valuable insights as well. Unstructured data is not useful when fit into a schema/table, unless there are specialized techniques that analyze some of the data and then store it in a column format.

Using the right tools, unstructured data can add a depth to data analysis that couldn’t be achieved otherwise. Structured data when enhanced from its unstructured data counterpart can provide a deeper insight.

\* <http://msrg.org/papers/Ghazal13>

TPCx-BB includes **Use Cases** based on the TPC-DS benchmark dealing with structured data, and adds Use Cases to address semi-structured and unstructured data in store and web sales channels. The semi-structured data is generated to represent the user clicks from a retailer's website to enable analysis of the user's behavior. This semi-structured data represent different user actions from a weblog and therefore varies in format.

The clickstream log contains data from URLs which are extracted from a Web server log. Typically, database and Big Data systems convert the webserver log to a table with the following five columns (DateID, TimeID, SalesID, WebPageID, UserID). To ease testing, such a table is generated in advance eliminating the need to extract and convert the webserver log information.

The unstructured part of the schema is generated in the form of product reviews, which are, for example, used for sentiment analysis. Figure 1 shows product reviews in the right part and their relationship to Date, Time, Item, Users and Sales tables in the structured part. The implementation of the product reviews is a single table with a structure like (DateID, TimeID, SalesID, ItemID, ReviewRating, ReviewText).

* + 1. Queries

TPCx-BB features thirty complex **Queries**, ten of which are based on the TPC-DS benchmark, the others were developed for TPCx-BB. The **Queries** cover areas of Big Data Analytics **Use Cases** such as Merchandising Pricing Optimization, Product Return Analysis, Inventory Management, Customers and Product Reporting.

1. -- WORKLOAD AND EXECUTION

.

* 1. Benchmark Kit

This clause defines TPCx-BB Kit contents, its workload execution process, allowed modification by the test sponsor, and contents of the run report.

* + 1. Kit Contents

The TPCx-BB kit contains the following:

* TPCx-BB Specification document.
* TPCx-BB Users Guide (README.md) documentation.
* Configuration files to adapt important parameters to the **SUT.**
* Bash scripts which control the benchmarking execution.
* A driver written in Java and Bash which implements the high-level run logic, time measurement and result computation
* A set of bash scripts which are called by the driver to perform benchmark and **Query** operations.
* Reference result set from SF 1GB.
* Set of scripts to automate result verification, checks on result cardinality and report generation.
  + 1. Kit Usage

To submit a compliant TPCx-BB **Result**, the **Test Sponsor** is required to use the TPCx-BB kit as outlined in the TPCx-BB Users Guide (README.md) with the following two exceptions:

* The setting of Kit Parameters files specified in Clause 2.1.4.
* Test Sponsor Kit Modifications explicitly allowed by Clause 2.1.5.

If there is a conflict between the TPCx-BB Specification and the TPCx-BB kit, the TPCx-BB kit implementation prevails.

* + 1. Kit Run report

The output of the TPCx-BB kit is called the run report which includes the following:

* Version number of TPCx-BB kit
* The start, end and total elapsed times for the 3 tests (Clause 2.4.1) of the **Performance Run**.
* The start, end and total elapsed times for the 3 tests (Clause 2.4.1) of the **Repeatability** **Run**.
* The output from the validation test to ensure the validation test was successful on the **SUT** (Clause 4.1.2.1)
* The computed TPCx-BB Secondary Metrics (Clause 5.6) for the **Performance Run**.
  + 1. Kit Parameter settings

The following files and parameters defined in Clauses 2.1.4.2 through 2.1.5.1 control the kit parameters that may be set by the **Test Sponsor**.

Generic Benchmark parameters defined in Appendix D

**Query** parameters defined in Appendix C have been tested to provide results for SF1 and are expected to produce results for larger scale factor test runs. Test sponsor can make syntactic changes but no values can be changed.

Global parameters are engine specific. The **Test Sponsor** can set their own parameters and must disclose as part of **FDR**. For example, please see below.

1. The Hive Global parameter file is located under $Big-Data-Benchmark-for-Big-Bench/distributions/%distribution%/%version%/ hive/conf/engineSettings.%files% E.g. (Appendix E) shows an example of Hive engine parameters; however the list is not exhaustive.
2. Global **Framework** parameters for those **Frameworks** which do not use HIVE can place their engine specific Global parameter file under be $Big-Data-Benchmark-for-Big-Bench/ distributions/%distribution%/%version%/ %engine%/conf/enginesettings.%files%.

**Comment:** It is acceptable that some engines may require other mechanisms for configuring parameters (e.g., use of GUI console to set parameters). Use of such approaches to parameterize is permissible provided the parameters comply with benchmark requirements and any non-default option settings are disclosed as part of the **FDR**.

* + 1. Test Sponsor Kit Modifications

Test Sponsor modifications to the provided scripts and configuration files in the TPCx-BB kit to facilitate system, platform and **Framework** differences are allowed without TPC approval. The allowed Test Sponsor Modifications are as follows:

* Script changes necessary for the kit scripts to execute on a particular Operating System as long as the changes do not alter the execution logic of the script.
* **Query** specific optimization **Framework** parameters can be specified either by using Global parameters as defined in Clause 2.1.4.4, or in local settings files under $Big-Data-Benchmark-for-Big-Bench/ distributions/%distribution%/%version%/ %engine%/**Queries**/q%%/enginelocalsettings.%files%. Appendix F provides an example of how these parameters can be defined.
* Custom metastore population scripts which can be passed using “-v” or placed under Big-Data-Benchmark-for-Big-Bench/ distributions/%distribution%/%version%/ %engine%/population/ and disclosed in the **FDR**.
* For non-hive **Frameworks,** custom engine settings can be passed using “-z”, or place it under Big-Data-Benchmark-for-Big-Bench/ distributions/%distribution%/%version%/ %engine%/conf/enginesettings.conf and disclosed in the **FDR.**

No modifications are allowed to the Java code provided in the TPCx-BB kit.

No JAR file optimizers are allowed to be used.

Any kit modifications not specified in Clause 2.1.5.1 must be brought forward to the Subcommittee as specified in Clause 2.2.

* 1. Benchmark Kit Modifications

For kit changes or modifications other than those allowed by Clause 2.1.4 and Clause 2.1.5 any TPC Member, company or individual may bring forward proposed kit changes to the TPCx-BB Benchmark Subcommittee. There are two methods of bringing forward these proposed kit changes.

Direct Method – A TPC Member, company, or individual may propose kit changes directly to the TPCx-BB Subcommittee.

Indirect Method – If the TPC Member, company, or individual wishes to remain anonymous then a **TPC Certified Auditor** can be used as an intermediary to interact with the TPCx-BB Subcommittee.

Regardless of which method is used the individual that will be interacting with the TPCx-BB Subcommittee becomes the Change Sponsor.

* + 1. Simple Review of Kit Modifications

For Third Tier (Clause 2.2.4.4) or Minor kit (Clause 2.2.4.2) modifications, the Change Sponsor shall present the proposed changes to the Subcommittee. The Subcommittee through its normal course of business will review the proposed changes, make the appropriate kit changes and bring forward the changes to the Council as a new revision of the TPCx-BB Benchmark.

If the proposed changes are significant, the Subcommittee may require that the Change Sponsor follow the Formal Review Process defined in Clause 2.2.2.

* + 1. Formal Review of Kit Modifications

For Major (Clause 2.2.4.1) kit Modifications, at the request to the Subcommittee or if the Change Spsonsor so desires, the Change Sponsor shall adhere to the following Formal Review Process.

* + - 1. **Formal Proposal of Kit Modifications**

Step 1: The Change Sponsor must submit to the chair of the TPCx-BB Subcommittee the following information:

* The proposed code changes or new **Framework** code
* The reason for proposing the changes
* Result set from the proposed changes
* Complete source code access if the proposed change prototype is available

To facilitate decision making process change sponsor may provide hardware and software required to validate and review the proposed changes.

Step 2: The chair of the TPCx-BB Subcommittee will add a discussion of the proposed changes to the agenda of the next Subcommittee meeting that can be attended by the Change Sponsor.

Step 3: The Change Sponsor will present the proposed changes to the TPCx-BB Subcommittee.

Step 4: The TPCx-BB Subcommittee will vote on one of three courses of action for the proposed changes.

* 1. Reject the proposed changes.
  2. Review the proposed changes as a Minor Kit Modification.
  3. Review the proposed changes as a Major Kit Modification.

If the proposed changes are rejected, no further action is necessary. Otherwise, the proposed changes immediately enter a Proposed Change Review period.

* + - 1. **Formal Review of Proposed Major Kit Modifications – Approximately six to twelve Week review period.**

If the proposed changes were voted to be a Major Kit Modification, then the Subcommittee chair will select at least three members of the Subcommittee to act as primary reviewers of the proposed changes. The Subcommittee chair will also determine the length of the review period and communicate the due date to the primary reviewers and to the Subcommittee. The primary reviewers' job is to examine and test the proposed changes. The primary reviewers are to give their recommendation to the Subcommittee no later than the due date set by the Subcommitte chair which is approximately six to twelve weeks.

* + - 1. **Formal Review of Proposed Minor Kit Modification – Six-week review period**

If the proposed changes were voted to be a Minor Kit Modification, then the Subcommittee chair will select at least two members of the committee to act as primary reviewers of the proposed changes. The primary reviewers job is to examine and test the proposed changes. The primary reviewers are to give their recommendation to the committee no more than six weeks later.

* + - 1. **Formal Rewiew by Subcomittee**

Once the review period ends and the primary reviewers have given their recommendations, the subcommittee will vote on whether to accept the proposed changes into the TPCx-BB benchmark kit.

If the changes are accepted, then the changes will be added to the kit.

* + 1. Kit Validation

Before any kit can be submitted for approval as a new revision of the TPCx-BB Benchmark Standard, all changes must pass the self-validation tests in the kit.

* + 1. Classification of Major, Minor and Third Tier Kit Modifications

It is necessary to ensure that the kit remains in sync with fast changing industry and technology land scape. The guidelines below illustrate the current structure of the Kit and help the Subcommittee to make a decision in a timely manner when evaluating a change proposal. These guidelines will help the Subcommittee do its due diligence and use its discretion to classify and process the change proposals. Modifications to the kit are divided into three types that follow the Revision classifications defined in the TPC Policies.

* + - 1. **Major Kit Modifications:**

Major Kit Modifications result in a significant change to the **Use Cases** or intent of the TPCx-BB Benchmark as to make **Results** from the new version non-comparable with the **Results** of the current TPCx-BB version.

These are a few examples of Major Kit Modifications:

* additions, deletions, and modifications to a **Use Case**
* changes to the Primary **Benchmark Metric**
* changes which may alter the reference result set
* changes made to run rules and Benchmark execution process
  + - 1. **Minor Kit Modifications:**

Minor Kit Modifications do not significantly alter the reference result set, the primary benchmark metrics, or the **Use Case**. Results are still comparable to the prior version. A few examples of Minor Kit changes:

* addition of a new **Framework** support
* bug fixes throughout the entire kit
* optimizations to the **Framework** specific code
* feature additions to Benchmark Driver
* modifications to tuning parameter files
* reference result set changes due to bug fixes
* **Framework** feature support
* updates to independent library files
* changes to the Data generator to support features and bugfixes
  + - 1. **Queries that use machine learning techniques**

Queries that use machine learning techniques (clustering or classification) don’t have a known correct answer set and so some other criteria must be applied to determine whether modifications are yielding **Results** that should be considered comparable. There are two general categories of changes that could impact the machine learning **Queries**:

1. Changes to the version/implementation of the SUT’s machine learning library (for example a new version of the Spark MLLib library) without any changes to kit itself. The concern in this case is that a new version of the machine learning library could make a different tradeoff in accuracy vs performance compared to earlier versions. The following criteria will be applied to evaluate whether results using a new library version should be comparable to previous **Results**:

* Results using the new library version must be generated without any changes to code or parameters in the kit (in particular there can be no changes to the input data, the parameters to the algorithm (e.g. number of iterations, number of clusters for KMeans, algorithm initialization parameters including seeds for any random initialization, regularization parameters for classification algorithms, etc).
* Results should only be considered comparable if the accuracy/evaluation metrics reported by the **Queries** are comparable. For example, the clustering **Queries** report the sum of squared distances from cluster centers as an accuracy metric, and the classification **Queries** report precision and AUC metrics. These metrics must demonstrate a level of accuracy for the new library implementation that is at least as good (within margin of error) as the accuracy of the earlier library version used in the comparison.

1. Introduction of new machine learning **Frameworks** (not just new versions of the previously supported framework) that may require actual changes in the kit code or parameters. This case is more subjective, but the general guidelines for considering results from a new ML **Framework** to be comparable are:
   * + The same input data must be used
     + To the extent that the new framework accepts similar parameters to existing frameworks (number of iterations, number of clusters, regularization parameters), the values for these parameters should be similar to those used for existing frameworks. If there is a need for the parameters to be different there must be sufficient technical justification provided.
     + The new **Framework** should be initialized using techniques that are comparable to the existing **Framework** (e.g. for clustering the new **Framework** should use the same random initialization approach).
     + The new **Framework** should be capable of reporting the same accuracy/evaluation metrics (sum of squared distance, precision, AUC, etc) as existing ML **Frameworks** and these metrics must demonstrate a level of accuracy for the new framework that is at least as good (within margin of error) as the accuracy of the earlier **Framework** used in the comparison.
       1. **Third Tier Kit Modifications:**

Third Tier Kit Modifications are those changes that clarify some confusing or ambiguous area of the kit, but do not alter the kit code or the **Use Cases**. Results are still comparable to the prior version. These are a few example of Third Tier changes:

* changes in documentation
  1. Benchmark Run

A valid run consists of 3 separate tests run sequentially. These tests may not overlap in their execution times. For example, the start of Test 2 may not begin until Test 1 is complete, the start of Test 3 may not begin until Test 2 is complete, etc. All tests are initiated by the TPCx-BB master scripts which can be executed from any of the nodes in the **SUT**. The tests are listed below:

* Load Test
* Power Test
* Throughput Test

The **Test Sponsor** sets the Benchmark Driver Parameters used during the tests are set per Clause 2.1.4.2.

The elapsed time for each test in Clause 2.3 must be reported.

Parameters *BENCHMARK\_START* and *BENCHMARK\_STOP* in TPCxBB\_Benchmarkrun.sh determine the overall elapsed time for the benchmark run.

**Test Database** is the database used to execute the database load test, Power test and Throughput test.

Database Location is the location of loaded data that is directly accessible (read/write) by the **Test Database** to perform the Load Test, Power Test and Throughput test.

Benchmark run should successfully pass Output data validation test as defined in Clause 4.1.2.9

* + 1. Load Test

The process of building the Test database is known as the Load Test. Database load consists of the following phases:

* + - 1. **Data Generation**: The process of using **PDGF** to create the data in a format suitable for presentation to the load facility. **PDGF** generates the data in a text-based flat file format and the flat files may be generated either:

1. to some location external to the **SUT.**
2. directly to some location on the **SUT** other than the final Database Location.
3. directly to the final Database Location.

If **PDGF** generates data directly into the final **Test Database** Location on the **SUT** (Clause 2.3.2.1 c) and the queries are subsequently run directly against the data in this location, then the generation and loading occur concurrently and both contribute to the database load time. If **PDGF** generates data to some location other than the final **Test Database** location (Clause 2.3.2.1 (a) or (b)), then the generation time is not included in the load time.

**Comment**: If a location external to the **SUT** is used for data generation, one should review Clause 6.2 to determine if there are implications on pricing.

Relocation: Copy to final Database Location. If the location of the **PDGF** output is different from the final Database Location, the data must be copied into the final Database Location. This phase is timed and contributes to the load time. Note that this copy may be done as part of the optional format conversion in the Data Preparation phase, in which case the time is captured as part of the Data Preparation timing. If multiple data copies occur between the **PDGF** generation and the placement of the data in the final Database Location, only the final copy into the Database location is included in the load time. For example, if **PDGF** generates data initially to a location external to the **SUT**, the flat files are subsequently copied to a staging area on the **SUT**, and then the data is copied again from the staging area into the Database Location as part of the Data Preparation format conversion, only the final copy is included in the load time.

* + - 1. Data preparation: The data preparation phase includes all additional work, beyond the Generation and Relocation steps, required to prepare the data for query execution. This includes the following steps:
* Creation of **Metadata** and population of the **Metastore**.
* Computing statistics for the database.
* Conversion of the data into an alternative or optimized format. An example would be conversion from the row-oriented format in the flat files to a compressed and/or columnar format. Note this is an optional step – if the flat files have been placed in the final Database Location by earlier load steps, then it's permissible to run queries directly against the flat files in their original format in the Database Location.

Any format conversion or creation of auxiliary data structures must meet the following requirements:

* it must not lose information from the original **Test Dataset**.
* it cannot make use of any knowledge of the **Queries** in the benchmark.

For example, the conversion can't remove columns that aren't referenced by the benchmark **Queries**, and creation of materialized views that pre-compute some or all of the query results is not allowed.

* + - 1. All work done during Data Preparation is timed and included in the load time.
    1. Power Test

Power test determines the maximum speed the **SUT** can process all 30 **Queries**. The **Queries** must run sequentially in ascending order.

* + 1. Throughput Test

Throughput Test runs 30 **Queries** using concurrent streams. Each stream runs all 30 **Queries** in a **Query** placement order mentioned in Clause 2.3.4.1. The Default streams for throughput test is set to 2, the number of concurrent streams are configurable with no maximum limit.

* + - 1. **Query placement in throughput test**

**Query** placement in the throughput test is perfomed using the automatic shuffling of the streams, Java code with the same seed is used in the driver to generate streams. **Query** placement for 100 streams are shown in Appendix J

* 1. Benchmark Execution

A Benchmark Execution is defined as a Validation test (Clause 4.1.2.1), Benchmark Run 1 followed by Benchmark Run 2.

Test sponsor runs the TPCxBB\_FullBenchmark\_sequence\_run.sh script. This script performs several steps in the following order.

* Step 1: Validation
* Step 2: First benchmark run (load, power and throughput)
* Step 3: Second benchmark run (load, power and throughput)

No activities except file system cleanup are allowed between Benchmark Run 1 and Benchmark Run 2. The **Performance Run** is defined as the run with the lower TPCx-BB **Performance Metric**. The **Repeatability Run** is defined as the run with the higher TPCx-BB **Performance Metric**. The **Reported Performance Metric** is the TPCx-BB **Performance Metric** for the **Performance Run**. There must not be any interruption during the tests, and all tests should be run without intervention.

No part of the **SUT** may be restarted during the Benchmark Execution. If there is a non-recoverable error reported by any of the applications, operating system, or hardware in any of the three tests (Clause 2.3 ) or between Run 1 and Run 2, the run is considered invalid. If a recoverable error is detected in any of the tests, and is automatically dealt with or corrected by the applications, operating system, or hardware, then the run is considered valid provided the run meets all other requirements. However, manual intervention by the **Test Sponsor** is not allowed. If a recoverable error requires manual intervention to deal with or correct, then the run is considered invalid.

* 1. Configuration and Tuning

The **SUT** cannot be reconfigured, changed, or re-tuned by the **Test Sponsor** during or between any of the three tests described in Clause 2.3. Any manual tunings to the **SUT** must be performed before the beginning of the benchmark execution described in Clause 2.4, and must be fully disclosed. Automated changes and tuning performed on the **SUT** between any of the tests are allowed. Any changes to default tunings or parameters of the applications, **Operating Systems**, or hardware of the **SUT** must be disclosed. Any changes deemed with the chracterstics of Benchmark Special in Clause 0.3.1 and Clause 0.3.2 are not allowed.

1. – System Under Test
   1. Logical Breakdown of System Under Test

The tested and **reported** configuration is composed of the hardware and software components that are employed in the TPCx-BB benchmark test and whose cost and performance are described by the benchmark metrics.

* + 1. System Under Test
* SUT can consist of Licensed Compute Services**.**
* Hardware component can be bare-metal, virtual machine or virtual instance.
* big Data Benchmark and Driver Software: TPCx-BB kit provides fully integrated benchmark and driver software to run on **SUT**.
* compute Software: Distributed compute software runs on Compute Hardware providing required software capabilities to successfully execute the benchmark.
* data Storage Software: Data Storage software runs on Data Storage hardware providing required software to create, store, and access input, output, intermediate, and temp data during the benchmark execution.
* compute Hardware: compute hardware provides multi-device compute capable hardware to execute the benchmark.
* data Storage Hardware: Data Storage hardware provides data storage capability using various kinds of persistient storage mediums.
* network Hardware and Software: Network Hardware and software provides capability to connect hardware and software in the **SUT** to communicate and perform data transfer over the network.
* devices in addition to listed above used in the **SUT**, for example compute devices and/or data storage devices, for e.g FPGA, Accelerator appliance, Accelerator cards, compression add-on cards, encryption add-on cards etc and their supporting software stack, device driver software, plug-in software.
* any hardware and software devices of all networks required to connect and support the **SUT** systems
* device running benchmark driver hardware and software resides on a separate system facilitating the execution of the benchmark, without interfering and influencing the **SUT**. This device is not part of the **SUT** and contains necessary SW and configuration to interact with the **SUT** and can be in form of Desktop, Notebook, or a Server.
* Figure 2 below shows an example **SUT** setup.

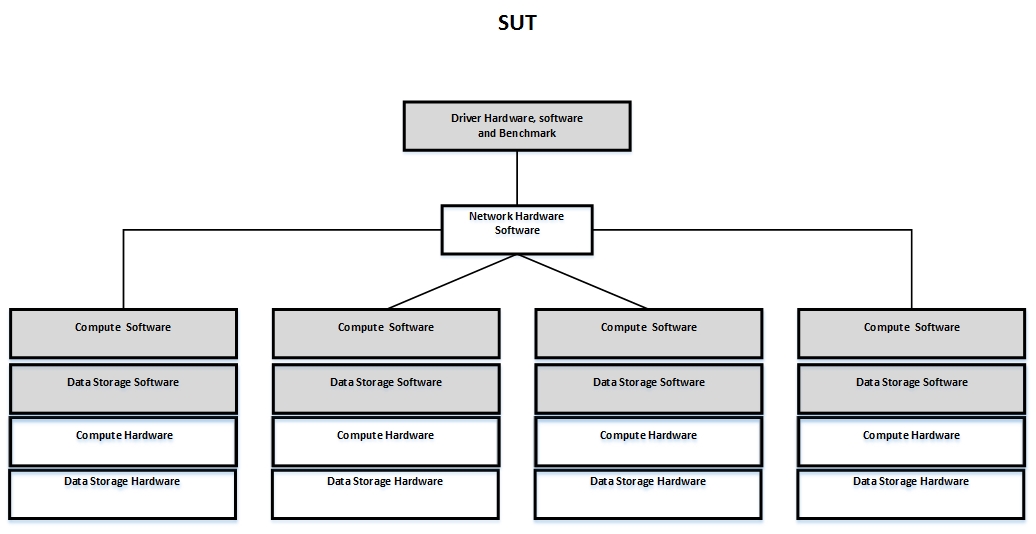


Figure 2 System under Test

* + 1. Commercially Available Products

Except for the TPCx-BB benchmark driver software, all **SUT** components must be **Commercially Available Products**. The source code of any non-commercially available products used to implement the **SUT** (including but not limited to scripts used to install, configure and tune the **SUT**) must be disclosed.

* + 1. Data Redundancy Requirement

The following clauses describe required **Data Redundancy** characterstics of the **SUT**. The failures described are not induced during the benchmark Execution.

Durable Medium: A durable medium that is either:

* 1. An inherently non-volatile medium (e.g., magnetic disk, magnetic tape, optical disk, solid state disk, Phase Change Memory, or techonolgy similar to Phase Change Memory. etc.) or;
  2. A volatile medium with its own self-contained power supply that will retain and permit the transfer of data, before any data is lost, to an inherently non-volative medium after the failure of external power.

The System Under Test must be configured to satisfy the requirements for **Data Redundancy** described in this clause. **Data Redundancy**, is demonstrated by the **SUT** being able to maintain operations with full data access during and after the permanent irrecoverable failure of any single storage Medium containing tables, input, output, or **Metadata** (including **MasterNode/NameNode** **Metadata** [where present]).

A configured and priced Uninterruptible Power Supply (UPS) is not considered external power.

DRAM can be considered a durable storage medium if it can preserve data long enough to satisfy the requirement (b) above. For example, if memory is accompanied by an Uninterruptible Power Supply, and the contents of memory can be transferred to an inherently non-volatile medium during the failure, then the memory is considered durable. Note that no distinction is made between main memory and memory performing similar permanent or temporary data storage in other parts of the system (e.g., disk controller caches).

* + - 1. Data Redundancy Reporting Requirements

The test sponsor must guarantee that the test system will not lose data due to a permanent irrecoverable failure of any single durable medium. Queries are not permitted to fail and no data can be lost in the presence of a single durable medium failure. This requirement applies to all Durable Medium containing TPCx-BB data (e.g., **Test Dataset** input data, table data, **EADS**, **Metadata** (including **MasterNode/NameNode** **Metadata** [where present], **Undo/Redo Log** data [where present], and “tempdb” [where present]). This requirement also applies to any benchmark query results(output data) stored on the SUT.

For data stored on distributed file systems (e.g., **HDFS)** providing redundancy via 3-way replication, erasure coding, etc:

1. Provide a report showing data resiliency. For **HDFS** this can be done by running “hdfs fsck –blocks”. “hdfs dfs -du -s –h” and “hdfs ec -getPolicy -path /”. When TPCxBB\_FullBenchmark\_sequence\_run.sh is run, this report will automatically be generated at the end of each benchmark run. (see Appendix H for sample output).

* For 3-way replication, the “default replication factor” should be at least 3 and “under-replicated blocks” should be zero.
* For erasure coding, the auditor must verify that the codec, node\_count, and parity settings results in redundancy at least equivalent to 3-way replication.

**Comment:** Typically this will be accomplished by verifying that “under-erasure-coded block groups”=0, num\_parity blocks ≥3 and node\_count ≥ (num\_data\_blocks+num\_parity\_blocks) but the auditor may need to perform an additional, implementation specific review.

1. For distributed file systems other than **HDFS,** redundancy has to be proved by the test sponsor. They must provide a description of the data redundancy approach describing both hardware and software used to achieve the data redundancy and explain why it is equivalent to the data redundancy provided by traditional local-**JBOD** storage and **HDFS** replication factor of at least 3.

**Comment:** If stored in a distributed filesystem, **Test Dataset** Input Data, **Metastore** Data and Output Data must be set to at least an equivalent of replication factor three for **HDFS** on **JBOD.** Non **HDFS** distributed file systems must demonstrate data redundancy equivalent to using replication factor three in **HDFS** as defined in Clause 3.1.3.3b.

For SUT components NOT using a distributed file system (e.g., **HDFS**) that provide redundancy via a **High Availability System:**

1. The test sponsor needs to provide a report that explains the configuration in sufficient detail to satisfy the auditor/PPB that outlines the use of distinct durable mediums for the individual service instances in the **HAS**.
2. While encouraged, there is no requirement for triple redundancy for this class of data.

Comment: A single durable medium failure could take down a service instance in the **HAS** but continued execution would be guaranteed by the existence of a secondary service instance using a distinct durable medium. In cases in which continued execution cannot be guaranteed when a service instance is taken down, the sponsor needs to implement other solutions to guarantee continued execution.

**Comment:** For consistency with the distributed file system model, no explicit test is necessarily required.

For SUT components that do NOT use a distributed file system (e.g., **HDFS**) or a **High Availability System,** (e.g., conventional direct attached storage on a single server):

1. The solution must guarantee uninterrupted access to the data on durable medium when a single Durable Media failure occurs.
2. The test sponsor must provide a report from a system tool detailing the media redundancy hardware/software configuration to the satisfaction of the auditor/PPB (e.g., a report showing that RAID-5 or RAID-10 is used).
3. While encouraged, there is no requirement for triple redundancy for this class of data.

**Comment:** Roll-forward recovery from an archive database copy (e.g., a copy taken prior to the run) using **Undo/Redo Log** data is not acceptable as the recovery mechanism in the case of durable medium failure. Note that “checkpoints”, “control points”, “consistency points”, etc. of the database taken during a run are not considered to be archives.

**Comment:** For consistency with the distributed file system model, no explicit test is necessarily required.

**Comment: Queries** may not fail due to a permanent irrecoverable failure of any single durable medium containing TPCx-BB data. However, medium failures are not allowed during benchmark runs to be considered valid (e.g., to avoid the possibility of higher performance when 3-way replication degrades into 2-way replication on medium failure).

**Comment:** At the heart of this requirement is handling the failure of any single durable media for consistency with other TPC benchmarks. For distributed file systems, while **HDFS** 2-way replication would have satisfied the spirit of this requirement, when TPCx-BB was created, **HDFS** deployments using 3-way replication were the norm (both for redundancy and for performance). Consequently, while the requirement mandates handling only a single point of failure, solutions using distributed file systems additionally require equivalence to HDFS with at least 3-way replication.

1. -- SCALE FACTORS and Result validation
   1. Scale Factor

The TPCx-BB benchmark defines a set of discrete scaling points (“scale factors”) based on the approximate size of the raw data produced by the datagenerator in Gigabytes.

Each defined scale factor has an associated value for SF, a unit-less quantity, roughly equivalent to the number of gigabytes of data present on the storage. The relationship between scale factors approximate size in Gigabytes is summarized in the Table 1 below.

Test sponsors may choose any scale factor from the defined series except SF1 which is used for **Result** validation only. No other scale factors may be used for a TPCx-BB **Result**.

|  |  |
| --- | --- |
| Scale Factor (SF) | Approximate Size in GB |
| 1 | 0.9 |
| 1000 | 923 |
| 3000 | 2794 |
| 10000 | 9450 |
| 30000 | 28740 |
| 100000 | 96923 |
| 300000 | 292792 |
| 1000000 | 989482 |

Table 1- Scale Factors

(Table 2-1) provides **Test Dataset** table sizes for each permissible scale factor.

**(**Table 2-2) provides **Test Dataset** table row counts for for each permissible scale factor.

The **Test Dataset** size (Table 2-1) information provided is an estimate and may vary from one benchmark submission to another depending on the data storage compression methods used. The estimate is provided solely to assist benchmark sponsors in the sizing of benchmark configurations. The datagenerator uses linear, log 1.5, log5, and sqrt scaling, depending on individual tables. The ratio of scaling between nominal scaling and generated data for a given SF is approximately 1.0.



Table 2-1 Dataset table sizes

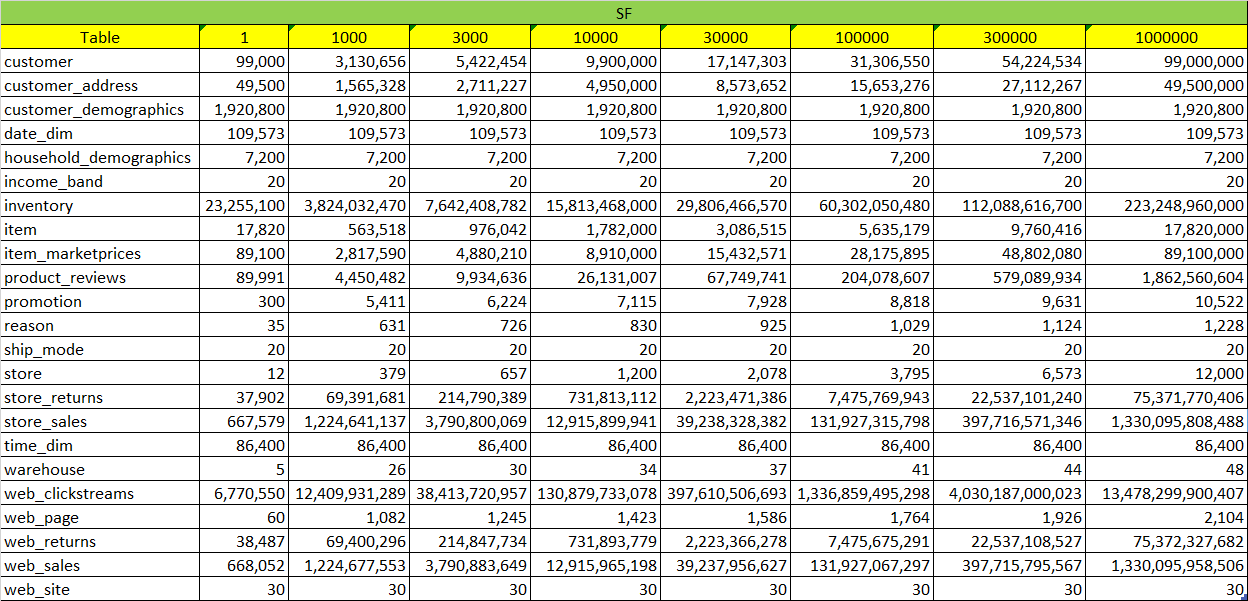


Table 2-2 Dataset Row count

* + 1. Result Validation

Result validation in TPCx-BB is performed as part of Validation Test in Clause 4.1.2.1 for **SUT** validation and Output validation for Run 1 and Run 2 in Benchmark Execution (Clause 2.4.1).

* + - 1. The validation test

The Validation performs data generation, load, power and validation run with scale factor 1 to perform an exact result validation against the reference result set in the kit . Validation test ensures that the engine used by the Test Sponsor to produce the publication can match the reference result set generated.

The validation result set for SF1 is the reference result used to validate the **SUT** for result correctness.

The intent of result validation is to validate **Queries** against SF1 and compare it against the reference result set packaged with the benchmark kit. This is the exercised against the **SUT** before the Benchmark Run 1 as part of **SUT** Validation Test.

Populate the **SUT** with SF1 dataset and schema information.

Execute the **Queries** using standard **Query** parameters as defined in the Queryparameters.sql (Appendix C) Verify the report generated by the driver matching the output to the reference result set.

Result Validation Process is defined in TPCXBB\_Validation.sh script and the generated report shall state that the output matches the reference result set for all 30 **Queries** the steps are provided below.

* *ENGINE\_VALIDATION\_DATAGENERATION*: This phase as defined in TPXBB\_Validation.sh generates a dataset at a fixed scale factor for 1 (SF1).
* *ENGINE\_VALIDATION\_LOAD\_TEST*: During this phase, as defined in TPXBB\_Validation.sh the data generated (Clause 4.1.2.4) will be loaded into the metastore.
* *ENGINE\_VALIDATION\_POWER\_TEST*: During this phase as defined in TPXBB\_Validation.sh, all 30 **Queries** will be run in sequence and the results are stored in the HDFS storage.
* *ENGINE\_VALIDATION\_RESULT\_VALIDATION*: During this automated phase as defined in TPXBB\_Validation.sh, the benchmark driver compares the results from all 30 **Queries** against a known reference results packaged with the kit.

The elapsed time for Validation Test is not included as part of Benchmark Metric calculation.

The elapsed time for Validation Test is not counted as part of Benchmark Execution.

For all other scale factors, used in the Run 1 and Run 2 (Clause 2.4.1) the benchmark driver at the end of the benchmark performs output validation checking for the presence of output data from power test and throughput test in order to qualify successful benchmark execution.

* + 1. Output data for Validation test.

After the execution of validation test, a **Query** returns one or more rows. The rows are called the output data.

Output data shall adhere to the following guidelines.

Output appears in the form specified by the **Query** description for **Queries** outputting data from SQL and procedural jobs.

Data will be formatted using the TPCx-BB result validation tool.

Non-integer numbers are expressed in decimal notation with two digits behind the decimal point.

Software library versions for Natural Language Processing are defined below and must be used in the result validation.

|  |  |  |
| --- | --- | --- |
| Library | Distro | Version |
| OpenNLP Tools | Apache | 1.9.3 |
| OpenNLP-Maxent | Sourceforge | 3.0.3 |

Strings are case-sensitive and must be displayed as such. Leading or trailing blanks are acceptable.

The amount of white space between columns is not specified.

The order of a **Query** output data must match the order of the validation output data, except for **Queries** that do not specify an order for their output data.

TPCx-BB automates result validation by strictly matching the outputed results for SF1 with reference resultset provided with the kit, the driver looks for exact result match. However, to accomodate situations where the validation results fails to match the reference result set, output data from a SF1 validation test shall adhere to the following rules to still qualify as valid test run.

* All tuples are part of the result and have the same values as reference result.
* An external post processing sorting with a bash script to bring the tuples into total global order for validation against the reference result set is acceptable.
* Use of the orderby feature, if supported to post process the data, is acceptable.
* For singleton column values and results from COUNT aggregates, the values must exactly match the corresponding values in the expected answer sets.
* For other numeric expressions including aggregates, the resulting values must be within 1% of the corresponding values in the expected answer sets.

Hadoop SQL **Frameworks** do not yet fully support the complete SQL standard. So does Machine learning libraries which are still evolving Clause 4.1.3.10 provide **Frameworks** to run a successful test where results are present but can not match strict reference result set due to missing feature and ordering.(E.g. Kit uses hive.optimize.sampling.orderby which is available in HIVE 0.14 but not in other versions).

1. Metrics
   1. TPCx-BB Primary Metrics

TPCx-BB defines the following primary metrics:

* BBQpm@SF, the **Performance Metric**, reflecting the TPCx-BB **Queries** per minute throughput; where SF is the Scale Factor (Clause 4.1)
* $/BBQpm@SF, the Price/**Performance Metric**
* **System Availability** Date as defined by the TPC Pricing Specification
* When the TPC-Energy option is chosen for reporting, the TPCx-BB energy metric reports the power per performance and is expressed as Watts/BBpm@SF. TPC Energy specification is located at [www.tpc.org](http://www.tpc.org). TPC-Energy metric is not mandatory.
  1. Performance Metric (BBQpm@SF)

The **Performance Metric** of the TPCx-BB benchmark, BBQpm@SF, is computed by combining metric components representing the load, power, and throughput tests.

* SF is the scale factor (4.1).
* TLD is the load factor computed as:

Where *TLoad* is the elapsed time of the Load Test (Clause 2.3.2) in seconds and 0.1 is a multiplication factor to adjust the contribution of Load test in the performance metric.

*TPT* is the geometric mean of the elapsed time Q in seconds of each of the 30 **Queries** as measured during the Power Test (2.3.3), multiplied by the number of **Queries** in the benchmark:

Where Q(*i*) is the elapsed time in seconds of **Query** *i* during the Power Test and M is the number of **Queries** in the Benchmark.

TTT  is the throughput test metric computed as the total elapsed time of the throughput test divided by the number of streams as measured during the Throughput Test (Clause 2.3.4).

*TTput* is the elapsed time of all streams from the Throughput Test.

* *n* is the number of streams in the Throughput Test (Clause 2.3.4).

Given the above definitions the overall **Performance Metric** BBQpm@SF is defined as:

Where *M* is the number of **Queries** in the Benchmark, SF is the Scale Factor and the factor of 60 in minutes in an hour.

* 1. Price Performance Metric ($/BBQpm@SF)

The **Price/Performance Metric** for the benchmark is defined as:

Where *C* is the total cost of ownership of the **SUT**.

If a benchmark configuration is priced in a currency other than US dollars, the units of the **Price/Performance Metrics** must be adjusted to employ the appropriate currency.

* 1. System Availability Date

The System **Availability Date** is defined in the TPC Pricing Specification.

* 1. Fair Metric Comparison

A TPCx-BB **Result** is only comparable with other TPCx-BB **Result**s of the same **Scale Factor** (Clause 4.1).

**Results** at different Scale Factors are not comparable, due to the substantially different computational challenges found at different data volumes. Similarly, the system **Price/Performance Metric** may not scale down linearly with a decrease in dataset size due to configuration changes required by changes in dataset size.

* 1. Secondary Metrics

Secondary metrics are additional metrics defined below are provided as part of the **Report**

* Computed Load Metric
* Computed Power Test Metric
* Computed Throughput Test Metric
* Elapsed time for each **Query** in Power test and Throughput test.

1. Pricing
   1. Introduction

This section defines the components, functional requirements of what is priced, and what **Substitutions** are allowed. How **Substitutions** are performed is defined in TPC Pricing Specification. Rules for pricing the **Priced Configuration** and associated software and maintenance are included in the TPC Pricing Specification located at [www.tpc.org](http://www.tcp.org/).

* + 1. Pricing Methodology

The Default 3-Year Pricing Methodology (as defined in the TPC Pricing Specification) must be used to calculate the price and the price/performance result of the TPCx-BB benchmark.

The Pricing Model 1 – Default Pricing Model (as defined in the TPC Pricing Specification) is the only pricing model allowed in a TPCx-BB result.

* 1. Priced Configuration

The system to be priced must include the hardware and software components present in the **System Under Test** (**SUT**), a communication interface that can support user interface devices, additional operational components con­figured on the test system, and maintenance on all of the above

Calculation of the priced configuration consists of:

* price of the **SUT** as tested and defined in Clause 3.1
* price of all software licenses for software used in the **SUT**
* price of a communication interface capable of supporting the required number of user interface devices defined in Clause 6.3
* price of additional products (software or hardware) required for customary operation, administration and maintenance of the **SUT** for a period of 3 years
* price of all products required to create, execute, administer, and maintain the executables or necessary to create and populate the test environment

**Comment**: Note that Clause 2.3.2.1 explicitly permits data generation to be external to the **SUT** in certain situations. In these situations the products required for such external to **SUT** data generation would not be priced if the auditor is satisfied that the solution meets the requirements of Clause 2.3.2.1.

Specifically excluded from the priced configuration calculation are:

* end-user communication devices and related cables, connectors, and switches.

end-user communication device here means driver node used to start, stop and orchestrate the benchmark, however devices used to connect to the end-user device with its cable is part of pricing.

* equipment and tools used exclusively in the production of the **Full Disclosure Report**
  1. Additional Operational Components

Additional products included on a customer installed configuration are also to be included in the priced configuration if explicitly required for the operation, administration, or maintenance of the priced configuration. Examples of such products are:

* operator console
* user interface terminal
* CD drive
* software, if required for initial load or maintenance updates
* all cables used to connect components of the **SUT** except as noted in Clause 6.2
  1. Allowable Substitutions

**Substitution** is defined as a deliberate act to replace components of the **Priced Configuration** by the **Test Sponsor** as a result of failing the availability requirements of the TPC Pricing Specification or when the part number for a component changes. This also requires compliance with the TPC Pricing Specification.

Corrections or "fixes" to components of the **Priced Configuration** are often required during the life of products. These changes are not considered **Substitutions** so long as the part number of the priced component does not change. Suppliers of hardware and software may update the components of the **Priced Configuration**, but these updates must not negatively impact the reported **Performance Metric** or numerical quantities more than two percent.

The following are not considered **Substitutions**:

* Software patches to resolve a security vulnerability
* Silicon revision to correct errors
* New supplier of functionally equivalent components (for example memory chips, disk drives, etc).

Some hardware components of the **Priced Configuration** may be substituted after the **Test Sponsor** has demonstrated to the **Auditor’s** satisfaction that the substituting components do not negatively impact the reported **Performance Metric** or numerical quantities. All **Substitutions** must be **Reported** in the **FDR** and noted by the **Auditor.** The following hardware components may be substituted:

* durable medium (for example disk drives) and cables
* durable medium enclosure
* network interface card
* router
* bridge
* repeater

If any hardware component is substituted, then the result must be audited by an Auditor (Clause 9.3.1).

1. – ENERGY

Energy metric in TPCx-BB is optional. For reporting requirements please refer to Energy Specification on

<http://www.tpc.org>

1. -- Full Disclosure Report
   1. Full Disclosure Report Requirements

A **Full Disclosure Report (FDR)** is required. This section specifies the requirements of the **FDR**.

The FDR is a zip file of a directory structure containing the following:

* a [Report](#report) in Adobe Acrobat PDF format
* an [Executive Summary Statement](#executive_summary_statement) in Adobe Acrobat PDF format
* The [Supporting Files](#supporting_files) consisting of any source files or scripts modified by the **Test Sponsor** and the output files generated by the TPCx-BB kit. Requirements for the FDR file directory structure are described below.

The **FDR** should be sufficient to allow an interested reader to evaluate and, if necessary, recreate an implementation of TPCx-BB **Result** given the appropriate hardware and software products. If any sections in the **FDR** refer to another section of the **FDR**, the names of the referenced scripts/programs must be clearly labeled in each section. Unless explicitly stated otherwise, “disclosed” or “reported” refers to disclosing or reporting in the **FDR**.

Since the test environment may consist of a set of scripts and corresponding input files, it is important to disclose and clearly identify, by name, scripts and input files in the **FDR**.

* 1. Format Guidelines

While established practice or practical limitations may cause a particular benchmark disclosure to differ from the examples provided in various small ways, every effort should be made to conform to the format guidelines. The intent is to make it as easy as possible for a reviewer to read, compare, and evaluate material in different benchmark disclosures.

All sections of the report, including appendices, must be printed using font sizes of a minimum of 8 points.

The **Executive Summary** must be included near the beginning of the **Report**.

The order and titles of sections in the Report and Supporting Files must correspond with the order and titles of sections from the TPCx-BB Specification (i.e., this document). The intent is to make it as easy as possible for readers to compare and contrast material in different Reports.

The directory structure of the FDR has three parts:

* ExecutiveSummaryStatement - contains Executive Summary statement
* Report - contains Report
* Supporting Files - contains Supporting Files
  1. General Items

The FDR must follow all reporting rules of the TPC Pricing Specification, located at [www.tpc.org](file:///C:\Users\mpoess.ST-USERS\AppData\Local\Microsoft\www.tpc.org). For clarity and readability, the TPC Pricing Specification requirements may be repeated in the TPCx-BB Specification.

A statement identifying the benchmark **Test Sponsor** and other participating companies must be provided.

Settings must be provided for all customer-tunable parameters and options that have been changed from the defaults found in actual products, including but not limited to:

* Configuration parameters and options for server, storage, network and other hardware components used by the **SUT**.
* Configuration parameters and options for the **Operating System** and file system components used by the **SUT**.
* Configuration parameters and options for any other software components (e.g. compiler optimization options) used by the **SUT**.

In the event that some parameters and options are set multiple times, it must be easily discernible by an interested reader when the parameter or option was modified and what new value it received each time.

This requirement can be satisfied by providing a full list of all parameters and options, as long as all those that have been modified from their default values have been clearly identified and these parameters and options are only set once.

Explicit response to individual disclosure requirements specified in the body of earlier sections of this document must be provided.

Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

* total number and type of nodes used
* total number and type of processors used/total number of cores used/total number of threads used (including sizes of L2 and L3 caches)
* size of allocated memory, and any specific mapping/partitioning of memory unique to the test;
* number and type of data storage units disk units, controllers, and if applicable, **LCS** volumes
* number of channels or bus connections to disk units, including their protocol type
* number of LAN (for example, Ethernet) connections and speed for switches and if applicable, other hardware components used in the test or are incorporated into the pricing structure
* type and the run-time execution location of software components

The following Figure 3 Sample Configuration Diagram illustrates a measured benchmark configuration where each server using Ethernet, an external driver, and two processors each with sixteen cores and two threads per core in the **SUT**. Note that this diagram does not depict or imply any optimal configuration for the TPCx-BB benchmark measurement.

Depending on the implementation of the **SUT**, the **Name Node**, Secondary **Name Node**, **Data Node**, Job/Task Tracker, Resource Manager/Node Manager, etc. or the functional equivalents must be specified in the diagram.



Figure 3 Sample Configuration Diagram

* n x Server Rack in scale out configuration.
* n x My Server Model B, 4/32/64 My CPU Model Z (2.7 GHz, 20MB cache, 130W), 128GB, My RAID Controller with 1GB BBWC
* n x My Storage Array Model A with 8 X 1TB 10K SAS HDD
* nx My Switch Model X 10GbE
* Nx Top of the Rack switch.
* **LCS** results can show LCS instance configuration instead of physical hardware equipment.

Detailed diagrams for system configurations and architectures can vary widely, and it is impossible to provide exact guidelines suitable for all implementations. The intent here is to describe the system components and connections in sufficient detail to allow independent reconstruction of the measurement environment. This example diagram shows homogeneous nodes. This does not preclude **Test Sponsors** from using heterogeneous nodes as long as the system diagram reflects the correct system configuration.

* 1. Software Components and Dataset Distribution

The distribution of software components, roles and dataset across all media must be explicitly described using a format similar to that shown in the following example for the tested and priced configuration.

Table 3 Example Layout Description

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Server** | **Role(s)** | **Count** | **Virtual** | **Host Name(s)** | **HW/SWConfiguration** | **Storage Setup** |
| Worker | Yarn NM/Hive Server/Spark Worker | 50 | N | TPCx-BB[100] -[BB150] | * Vendor Server Model Name. * HW/SW Config (Processor Model, socket count, Frequency, Core count). * DRAM capacity. * Storage x HDD Model. * Network and BW link speed. * OS Model and version. * Framework SW Model and version. * Details of Additional HW/SW if any. | **OS: Model x GB SSD, Intermediate/Shuffle/Temp Data: x Model x GB SSD,Distributed FS: x Model 12x SAS/SATA Harddrive/** |
|  |  |  |  |  |  |  |
| Distro Manger | Hadoop Manager | 1 | N | TPCx-BB-CDH | * Vendor Server Model Name. * HW/SW Config (Processor Model, socket count, Frequency, Core count). * DRAM capacity. * Storage x HDD Model. * Network and BW link speed. * OS Model and version. * Framework SW Model and version. * Details of Additional HW/SW if any. | **OS: Model x GB SSD.** |
|  |  |  |  |  |  |  |
| Gateway **SUT** Driver | YARN/SPARK,HIVE Gateway | 1 | N | TPCxBB-Driver 1 | * Vendor Server Model Name. * HW/SW Config (Processor Model, socket count, Frequency, Core count). * DRAM capacity. * Storage x HDD Model. * Network and BW link speed. * OS Model and version. * Framework SW Model and version. * Details of Additional HW/SW if any. |  |
|  |  |  |  |  |  |  |
| Name Node/Resource Manager | YARN/NN/ZooKeeper | 1 | N | TPCx-BB\_PNN1 | * Vendor Server Model Name. * HW/SW Config (Processor Model, socket count, Frequency, Core count). * DRAM capacity. * Storage x HDD Model. * Network and BW link speed. * OS Model and version. * Framework SW Model and version.   Details of Additional HW/SW if any. |  |
|  |  |  |  |  |  |  |

The distribution of various software components across the system must be explicitly described using a format similar to that shown in Table-3 in Clause 8.4 for both the tested and priced configuration.

Software components might vary from across different implementations.

The distributed file system implementation (for example Apache HDFS, Red Hat Storage, IBM GPFS, EMC Isilon OneFS) and corresponding Hadoop File System API version must be disclosed.

The Engine implementation (for example, Apache Map/Reduce, Spark, Flink, IBM Platform Symphony) and corresponding version must be disclosed.

**Frameworks** and Engine used in the benchmark should be disclosed in the report.

If there were any additional vendor supported patches applied to the **SUT**, details of such patches should be disclosed.

* 1. Workload Related Items

Script or text used to set for all hardware and software tunable parameters must be Included in the **Report**.

Version number of TPCx-BB kit must be Included in the **Report**.

The run report generated by TPCx-BB benchmark kit must be included in the **Report**.

Elapsed times of all power and throughput **Queries** needs to be reported from the **Performance Run**, grouped respectively as Structured, semi-structured and unstructured buckets. Must be included in the **Report** for groupings please see clause B.3 in Appendix B.

**Query** completion times for individual **Queries** run as part of **Performance Run** should be included in the **Report**.

Output report from successful **SUT** Validation test must be included in the **Report** (Clause 4.1.2.1)

Global **Framework** parameter settings files (Clause 2.1.4.4) must be included in the **Report**.

Test Sponsor Kit modification files (Clause 2.1.5) must be included in the **Report**.

* 1. SUT Related Items

Specialized Hardware/Software used in the **SUT** must be included.

All **Framework** configuration files from **SUT**, for the performance run E.g Yarn-Site.xml, Hdfs-site.xml etc.

**SUT** environment info in form of envinfo.log from a representative worker node from every role in the server. See (Appendix G)

The data storage ratio must be disclosed. It is computed by dividing the total physical data storage present in the **Priced Configuration** (expressed in TB) by the chosen Scale Factor as defined in Clause 4.1. Let r be the ratio. The **Reported** value for r must be rounded to the nearest 0.01. That is, reported value=round(r, 2). For example, a **SUT** configured with 96 disks of 1TB capacity for a 1TB Scale Factor has a data storage ratio of 96.

For the reporting of configured data storage capacity, terabyte (TB) is defined to be 10^12 bytes.

For consumption based storage provisioning in **LCS**, the maximum storage provisioned during the entire benchmark test is considerd to be the total physical data storage present.

The Scale Factor to memory ratio must be disclosed. It is computed by dividing the Scale Factor by the total physical memory present in the **Priced Configuration**. Let r be this ratio. The **Reported** ratio must be rounded to the nearest 0.01. That is, reported value=round(r,2). For example, a system configured with 1TB of physical memory for a 10TB Scale Factor has a memory ratio of 10.00.

For LCS, the maximum provisioned memory during the entire benchmark test is consided to be the total physical memory present.

* 1. Metrics and Scale Factors

The **Reported Performance Metric** (BBQpm@SF for the **Performance Run**) must be disclosed in the **Report**.

The **Performance Metric** (BBQpm@SF) for the **Repeatability Run** must be disclosed in the **Report**.

The **Price/Performance Metric** ($/BBQpm@SF) for the **Performance Run** must be disclosed in the **Report**.

The Scale Factor used for the **Result** must be disclosed in the **Report.**

The number of streams in the throughput run used for the **Result** must be disclosed in the **Report.**

The total elapsed time for the execution of the **Performance Run** and **Repeatability Run** must be disclosed in the **Report.**

The time for each of the three tests (Clause 2.4.1) must be disclosed for the **Performance Run** and **Repeatability Run.**

* 1. Audit Related Items

If the benchmark is audited by an independent **Auditor**, the **Auditor's** agency name, address, phone number, and **Attestation Letter** with a brief audit summary report indicating compliance must be included in the **Report**. A statement should be included specifying whom to contact in order to obtain further information regarding the audit process.

* + - 1. ExecutiveSummaryStatement

The **Executive Summary** is a high level overview of a TPCx-BB implementation. It should provide the salient characteristics of a benchmark execution (metrics, configuration, pricing, etc.) without the exhaustive detail found in the **FDR**. When the TPC-Energy optional reporting is selected by the **Test Sponsor**, the additional requirements and format of TPC-Energy related items in the **Executive Summary** are included in the TPC Energy Specification, located at [www.tpc.org](file:///C:\Users\mpoess.ST-USERS\AppData\Local\Microsoft\www.tpc.org).

The **Executive Summary** has three components:

* Implementation Overview
* Pricing Spreadsheet
* Numerical Quantities

Page Layout

Each component of the **Executive Summary** should appear on a page by itself. Each page should use a standard header and format, including:

* 1/2 inch margins, top and bottom
* 3/4 inch left margin, 1/2 inch right margin
* 2 pt. frame around the body of the page. All interior lines should be 1 pt.
  + 1. Implementation Overview

The implementation overview page contains five sets of data, each laid out across the page as a sequence of boxes using 1 pt. rule, with a title above the required quantity. Both titles and quantities should use a 9-12 pt. Times font unless otherwise noted.

The first section contains information about the sponsor and system identification.

Table 4 Sponsor and System Identification

|  |  |
| --- | --- |
| **Title** | **Font** |
| Sponsor Name or Logo | 16-20 pt. Bold (for Name) |
| System Identification | 16-20 pt. Bold |
| Version Numbers for TPCx-BB, TPC-Pricing and TPC-Energy (if reported) | 16-20 pt. Bold |
| Report Date | 16-20 pt. Bold |

It is permissible to use or include company logos when identifying the sponsor.

The report date must be disclosed with a precision of 1 day. The precise format is left to the **Test Sponsor**.

The second section contains the Total System Cost, the TPCx-BB **Reported** **Performance Metric** and the **Price/Performance Metric**.

Table 5 Benchmark Results

|  |  |  |  |
| --- | --- | --- | --- |
| **Title** | **Quantity** | **Precision** | **Font** |
| Total System Cost | 3 yr. Cost of ownership (Clause 6.2) | 1 | 16-20 pt. Bold |
| **Reported** **Performance Metric** | BBQpm (Clause 5.2) | 0.01 | 16-20 pt. Bold |
| **Price/Performance** | $/ BBQpm (Clause 5.3) | 0.01 | 16-20 pt. Bold |

Depending on the currency used for publication this $ sign mustbe changed to ISO currency symbol.

The third section contains detailed diagrams of the measured configuration (Clause 8.3.1.5) and the Software components distribution table (Clause 8.4)

Table 6 System Configuration Information

|  |  |  |
| --- | --- | --- |
| **Title** | **Quantity** | **Font** |
| **Framework** /Engine Software | Product name and Product Version | 9-12 pt. Times |
| **Operating System** | Product name, **Software Version** of OS, File System Type and Version | 9-12 pt. Times |
| Other Software | Product name and **Software Version** of other software components (example Java) | 9-12 pt. Times |
| System **Availability** **Date** | The **Availability Date** of the system, defined in the TPC Pricing Specification (Clause 6) | 9-12 pt. Times |
| SF (Scale Factor) | SF in as defined in Clause 4.1 | 16-20pt. Bold |
| Streams | Concurrent Streams used in Clause 2.3.4 | 16-20pt. Bold |

The **Software Version** must uniquely identify the orderable software product referenced in the **Priced Configuration** (for example, RALF/2000 4.2.1).

The fourth section contains the storage and memory ratios, see (Clause 8.6.)

Table 7 Storage and Memory Ratios

|  |  |  |
| --- | --- | --- |
| **Title** | **Precision** | **Font** |
| Physical Storage/Scale Factor | 0.01 | 9-12 pt. Times |
| Scale Factor/Physical Memory | 0.01 | 9-12 pt. Times |
| Main Data Redundancy Model | n/a | 9-12 pt. Times |

The fifth section contains the components, including:

* total number and type of nodes used/total number of processors used with their types and speeds in GHz/ total number of cores used/total number of threads used, see (Clause 8.3.1.5)
* main and cache memory sizes
* network speed and topology (e.g Top of the Rack switch, in-rack switch)
* storage type, quantity and configuration.
  + 1. Pricing Spreadsheet

The major categories in the Price Spreadsheet, as appropriate, are as follows:

* network(s)
* server(s) /node(s)
* storage
* software

Discounts (may optionally be included with above major category subtotal calculations).

* + 1. Numerical Quantities Summary

The Numerical Quantities Summary page contains three sets of data, presented in tabular form, detailing the Load Phase, Power Phase, and throughput phase execution timings for the **Performance Run** and **Repeatability Run**. Each set of data should be headed by its given title and clearly separated from the other tables.

The first section contains the Scale Factor, Number of streams, and **SUT** Validation test status for the reported benchmark execution **Result**.

The second section contains measurement results and metric from the **Performance Run**.

Table 8 Measurement Results for Performance Run

| **Performance Run** | |
| --- | --- |
| **Item Title** | **Precision** |
| Overall Run Start Time | yyyy-mm-dd hh:mm:ss.sss |
| Overall Run End Time | yyyy-mm-dd hh:mm:ss.sss |
| Overall Run Total Elapsed Time | ss.sss |
| Start of Load Test | yyyy-mm-dd hh:mm:ss.sss |
| End of Load Test | yyyy-mm-dd hh:mm:ss.sss |
| Load Test Elapsed Time | ss.sss |
| Start of Power Test | yyyy-mm-dd hh:mm:ss.sss |
| End of Power Test | yyyy-mm-dd hh:mm:ss.sss |
| Power Test Elapsed Time | ss.sss |
| Throughput Test | yyyy-mm-dd hh:mm:ss.sss |
| Throughput Test | yyyy-mm-dd hh:mm:ss.sss |
| Throughput Test Elapsed Time | ss.sss |
| Performance Metric (BBQpm@SF) | x,xxx.xx |

* + - 1. The third section contains the measurement results for the **Repeatability Run**. See Table 8 for contents and precision.
    1. TPCx-BB Run Report

The Run Report per Clause 2.1.3 from TPCx-BB must be included in the **Report** immediately after the **Executive Summary.**

* 1. Availability of the Full Disclosure Report

The **Full Disclosure Report** must be readily available to the public. The **Report** and Supporting Files must be made available when the **Result** is made public. In order to use the phrase “TPCx-BB Benchmark”, the **Full Disclosure Report** must have been previously submitted electronically to the TPC using the procedure described in the TPC Policies and Guidelines document.

The official **Full Disclosure Report** must be available in English but may be translated to additional languages.

* 1. Revisions to the Full Disclosure Report

The requirements for a revision to an **FDR** are specified in the TPC Pricing Specification.

1. – Auditing
   1. TPC Pricing

All audited requirements specified in the TPC Pricing Specification located at [www.tpc.org](http://www.tpc.org) must be followed. The TPCx-BB pricing information included in the **Report** must be audited by a TPC certified Auditor. Test Sponsor should submit the Pricing data specified in the version of TPC Pricing Specification located at [www.tpc.org](http://www.tpc.org).

* 1. Optional TPC-Energy Results

When the TPC-Energy optional reporting is selected by the **Test Sponsor**, the rules for auditing of TPC-Energy related items are included in the TPC Energy Specification located at [www.tpc.org](file:///C:\Users\mpoess.ST-USERS\AppData\Local\Microsoft\www.tpc.org). If TPC-Energy metrics are **Reported**, the TPCx-BB Energy results must be audited by a TPC-Energy certified **Auditor**.

* 1. General Rules

Before publication, a TPCx-BB **Result** must be certified to be compliant with the spirit and letter of the TPCx-BB Benchmark Standard by an Independent Certified TPC Auditor or a TPCx-BB Pre-Publication Board. The **Test Sponsor** can choose the certification be performed by either by a Certified TPC Auditor or a Pre-Publication Board.

1. * 1. Independent Audit

The term independent is defined as “the outcome of the benchmark carries no financial benefit to the auditing agency other than fees earned directly related to the audit.” The independent audit of the benchmark is described in TPC Policies on [www.tpc.org](http://www.tpc.org)

The **Auditor**’s opinion regarding the compliance of a **Result** must be consigned in an **Attestation Letter** delivered directly to the **Test Sponsor**. To document that a Result has been audited, the Attestation Letter must be included in the Report and made readily available to the public. Upon request, and after approval from the Test Sponsor, a detailed audit report may be produced by the Auditor.

* + 1. Pre-Publication Board

The term Pre-Publication Board as defined by the TPC Policies consists of one or more individuals that have been chosen by the TPCx-BB Benchmark Subcommittee to certify **Results** for publication. For TPCx-BB **Results** the Pre-Publication Board consists of 3 members from the TPCx-BB Benchmark Subcommittee. Each member serves a period of six months. The membership will be rotated through the TPCx-BB Benchmark Subcommittee membership. The submission is confidential to the Pre-Publication Board until the **Result** is published. The Pre-Publication Board must complete the review within 10 business days. If no issues are raised within the 10 business day period, the **Result** is considered certified for publication.

* + 1. Results Based on Existing TPCx-BB Results

A **Test Sponsor** can demonstrate compliance of a new **Result** produced without running any performance test by referring to the certification of a **Result**, if the following conditions are all met:

* The referenced **Result** has already been published by the same or by another **Test Sponsor**.
* The new **Result** must have the same hardware and software architecture and configuration as the referenced **Result**. The only exceptions allowed are for elements not involved in the processing logic of the **SUT** (e.g., number of peripheral slots, power supply, cabinetry, fans, etc.)
* The **Test Sponsor** of the already published **Result** gives written approval for its use as referenced by the **Test Sponsor** of the new **Result**.
* The **Auditor** verifies that there are no significant functional differences between the priced components used for both **Results** (i.e., differences are limited to labeling, packaging and pricing.)
* The **Auditor** or Pre-Publication Board reviews the **FDR** of the new **Result** for compliance.
* If certification is performed by an independent **Auditor**, a new **Attestation Letter** must be included in the **Report** of the new **Result**.

The intent of this clause is to allow publication of benchmarks for systems with different packaging and model numbers that are considered to be identical using the same benchmark run. For example, a rack mountable system and a freestanding system with identical electronics can use the same benchmark run for publication, with, appropriate changes in pricing.

Although it should be apparent to a careful reader that the **FDR** for the two **Results** are based on the same set of performance tests, the **FDR** for the new **Result** is not required to explicitly state that it is based on the performance tests of another published **Result**.

When more than one **Result** is published based on the same set of performance tests, only one of the **Results** from this group can occupy a numbered slot in each of the benchmark **Result** “Top Ten” lists published by the TPC. The **Test Sponsors** of this group of **Results** must all agree on which **Result** from the group will occupy the single slot. In case of disagreement among the **Test Sponsors**, the decision will default to the **Test Sponsor** of the earliest publication from the group.

* 1. Audit Checklist

A generic audit checklist is provided as part of this specification. The genericaudit checklist specifies the requirements that must be checked to ensure a Result is compliant with the TPCx-BB Specification. Not only should the TPCx-BB requirements be checked for accuracy but the Auditor or Pre-Publication Board must ensure that the FDR accurately reflects the Result.

An independent **Auditor** must be used for those audit checklist items that refer to pricing or energy.

Verify that the TPCx-BB provided kit is used and its version.

Verify that all 3 tests (Load, Power, Throughput) (Clause 2.3) of the **Performance Run** and **Repeatability Run** completed with no error reported.

Verify Validation tests (Clause 4.1.2.1) of **Performance Run** completed with no error reported.

Verify Benchmark Execution has been executed according to Clause 2.4.

Verify the Validation test results reported for SF1 matches with reference result set provided with the TPCx-BB kit (Clause 4.1.2.6) If the Validation test results do not match with the reference result set use manually verify validation test results as defined in Clause 4.1.3.10.

Verify that all scripts and source code to implement the benchmark has been included in the **Report**.

Verify Kit run-report contains all information mentioned in Clause 2.1.3.

Verify Clause 2.1.4 has been followed to ensure the parameter settings was performed as defined in the specification and required reports, files are provided as part of the FDR.

Verify Clause 2.1.5 is followed and according the defined Test Sponsor Kit modification.

Verify Clause 2.1.5.2 and 2.1.5.3 is followed and no Java code files were modified and no JAR file optimizers were used.

Verify the test execution has produced the required output by checking the logfiles to see if all the **Queries** have created an output.

Verify that all components of the **SUT** are commercially available as per the TPC Pricing Specification.

Verify that all components of the **SUT** are included in the pricing.

Verify no aspect of **SUT**, including the dataset size, tuning parameters were changed between the **Performance Run** and **Repeatability Run** .

Verify that the SF used for publication is valid according to Clause 4.1.

Verify that the metrics are **Reported** as per the requirements in Clause 5

Verify that the **SUT** Pricing **Report** is in compliance with the TPC Pricing specification.

**Comment:** The auditor should also review the **SUT** pricing details in Clause 6.2 as they verify the **SUT** Pricing **Report**

Verify that the Energy report is in compliance with the TPC Energy specification (if reported).

Verify that Full Disclosure Report and Executive Summary Reports are accurately reported and comply with the reporting requirements. This includes but not limited to.

* metric calculation
* system availability
* the diagrams of both measured and priced configuration
* system pricing
* the numerical quantity summary
* Parameter files required as part of **FDR** are provided.

1. **Sample Executive Summary**

The following page provides a template of the TPCx-BB **Executive Summary****.**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| My Company Logo | | | My Server/LCS Model B | | | | TPCx-BB Rev. 1.1.0  TPC-Pricing Rev. 2.0.1 | | |
| Report Date: December 15, 2014 | | |
| Total System Cost | | | Performance Metric | | | | Price / Performance | | |
| $99,996.13 USD | | | 390.99 BBQpm@3000 | | | | $255.76 USD $ / BBQpm @3000 | | |
| Scale Factor | Streams | Apache Hadoop Compatible software | | | Operating System | | Other Software | | Availability Date |
| 3000 | 4 | My HDFS Software 1.0 | | | My OS V2.0 | | None | | December 15, 2014 |
| System Configuration | | | | | | | | | |
| Physical Storage /Scale Factor:250 | | | | Scale Factor/Physical Memory: 5.8 | | | | Main Data Redundancy Model: 3-way replication | |
| Servers/LCS | | | | | | 4 x My Server Model B | | | |
| Processors/Cores/Threads/Model | | | | | | 4/32/64 My CPU Model Z (2.7 GHz, 20MB cache, 130W) | | | |
| Memory | | | | | | 128GB | | | |
| Storage | | | | | | 2 x 600GB 10K SFF SAS (internal)  1 x My Storage Array Model A with 8 X 1TB 7.2K SAS LFF HDD | | | |
| Network: | | | | | | 2x My Switch Model X 10GbE | | | |
|  | | | | | |  | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| My Company Logo | My Server/LCS Model B | | | | | TPCx-BB Rev. 1.1.0 | | |
| TPC-Pricing Rev. 2.0.1 | | |
| Report Date: | | 15-Dec-2014 |
| Description | Part Number | |  | Source | Unit Price | Qty | Extended Price | 3 Year Maint. Price |
| My Server/LCS Model B, 4 My CPU Model Z, 128GB, 2 x 600GB 10K SFF SAS | MY-S-001 | |  | 1 | 12,100.77 | 4 | $48,403 | $100 |
| My Storage Array Model A | MY-SE-002 | |  | 1 | 1,988.00 | 4 | $7,952 | $200 |
| My HDD Model xyz 1TB SATA 7.2K LFF | MY-HDD-011 | |  | 1 | 800.47 | 40 | $32,019 |  |
| My OS | MY-OS | |  | 1 | 485.24 | 4 | $1,941 |  |
| My HDFS SoftwareSoftware | MY-Hadoop | |  | 1 | 2,700.00 | 4 | $10,800 |  |
| My Switch Model X | My-Switch | |  | 1 | 1,922.12 | 2 | $3,844 |  |
|  |  |  | |  |  | Subtotal | $104,959 | $300 |
| Large Purchase Discount | 5.0% |  | | 1 |  |  | -$5,248 | -$15 |
| Pricing: 1=My Company | | | | | Three-Year Cost of Ownership: | | | $99,996.1 |
| Audited by My Auditor | | | | |
| All discounts are based on US list prices and for similar quantities and configurations. The discounts are based on the overall specific components pricing from respective vendors in this single quotation. Discounts for similarly sized configurations will be similar to those quoted here, but may vary based on the components in the configuration. | | | | |  |  | BBQpm@SF | 1,100.1 |
|  |  |  |  |
|  |  | $/BBQpm: | $90.9 |
| Prices used in TPC benchmarks reflect the actual prices a customer would pay for a one-time purchase of the stated components. Individually negotiated discounts are not permitted. Special prices based on assumptions about past or future purchases are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details, see the pricing sections of the TPC benchmark specifications. If you find that the stated prices are not available according to these terms, please inform at pricing@tpc.org. Thank you. | | | | | | | | |

|  |  |  |
| --- | --- | --- |
| My Company Logo | My Server/LCS Model B | TPCx-BB Rev. 1.1.0  TPC-Pricing Rev. 2.0.1 |
| December 15, 2014 |
| |  |  | | --- | --- | | Measurement Results | | | Scale Factor | 3000 | | Number of Streams | 4 | | **Performance Run**  **S** |  | | Start of Validation Test | 10/02/2014 01:02:09.123 | | End of Validation Test | 10/02/2014 01:15:56.676 | | Validation Test Result | Success | | Start of Run | 10/02/2014 02:01:09.342 | | End of Run | 10/02/2014 08:11:31.765 | | Run Elapsed Time | 6:10:22.342 | | Start of Load Test | 10/02/2014 02.01:09.376 | | End of Load Test | 10/02/2014 02:01:16.326 | | Load Test Elapsed Time | 3:10:22.654 | | Start of Power Test | 10/02/2014 03:08:26.328 | | End of Power Test | 10/02/2014 03.08:27 | | Power Test Elapsed Time | 3:10:22.373 | | Start of Throughput Test | 10/02/2014 03:08:26.273 | | End of Throughput Test | 10/02/2014 03.08:27.235 | | Throughput Test Elapsed Time | 3:10:22.234 | | Performance Metric (BBQpm@SF) | 398.99 @ BBQpm SF |      |  |  | | --- | --- | | **Repeatability Run** |  | | Start of Validation Test | 10/02/2014 01:02:09.123 | | End of Validation Test | 10/02/2014 01:15:56.676 | | Validation Test Result | Success | | Start of Run | 10/02/2014 02:01:09.342 | | End of Run | 10/02/2014 08:11:31.765 | | Run Elapsed Time | 6:10:22.342 | | Start of Load Test | 10/02/2014 02.01:09.376 | | End of Load Test | 10/02/2014 02:01:16.326 | | Load Test Elapsed Time | 3:10:22.654 | | Start of Power Test | 10/02/2014 03:08:26.328 | | End of Power Test | 10/02/2014 03.08:27 | | Power Test Elapsed Time | 3:10:22.373 | | Start of Throughput Test | 10/02/2014 03:08:26.273 | | End of Throughput Test | 10/02/2014 03.08:27.235 | | Throughput Test Elapsed Time | 3:10:22.234 | | Performance Metric (BBQpm@SF) | 398.99 @ BBQpm SF | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | | | |

1. Logical Database Design

The following Appendix provides an overview of the data model and all table columns implemented by the TPCx-BB kit. If there is a conflict between the descriptions provided in this TPCx-BB Specification and the TPCx-BB kit implementation, the TPCx-BB kit prevails.

* 1. Table Columns Used by Queries

Minimal data description (contains only columns used by **Queries**) (~122 columns).

**date\_dim**

|  |  |  |  |
| --- | --- | --- | --- |
| date\_dim | Type | NULL? | Table is used by **Queries**: |
| d\_date\_sk | BIGINT | NOT NULL | Q4 Q6 Q7 Q9 Q13 Q16 Q17 Q19 Q21 Q22 Q23 |
| d\_date | DATE |  | Q4 Q16 Q19 Q22 |
| d\_month\_seq | INTEGER |  | Q7 |
| d\_week\_seq | INTEGER |  | Q19 |
| d\_year | INTEGER |  | Q6 Q7 Q9 Q13 Q17 Q21 Q23 |
| d\_moy | INTEGER |  | Q7 Q17 Q21 Q23 |

**time\_dim**

|  |  |  |  |
| --- | --- | --- | --- |
| time\_dim | Type | NULL? | Table is used by **Queries**: |
| t\_time\_sk | BIGINT | NOT NULL | Q4 Q14 |
| t\_time | INTEGER |  | Q4 |
| t\_hour | INTEGER |  | Q14 |

**customer**

|  |  |  |  |
| --- | --- | --- | --- |
| Customer | Type | NULL? | Table is used by **Queries**: |
| **c\_customer\_sk** | BIGINT | NOT NULL | Q5 Q6 Q7 Q13 Q17 |
| c\_customer\_id | CHAR (16) | NOT NULL | Q6 Q13 |
| c\_current\_cdemo\_sk | BIGINT |  | Q5 |
| c\_current\_addr\_sk | BIGINT |  | Q7 Q17 |
| c\_first\_name | CHAR (20) |  | Q6 Q13 |
| c\_last\_name | CHAR (30) |  | Q6 Q13 |
| c\_preferred\_cust\_flag | CHAR (1) |  | Q6 |
| c\_birth\_country | VARCHAR (20) |  | Q6 |
| c\_login | CHAR (13) |  | Q6 |
| c\_email\_address | CHAR (50) |  | Q6 |

**customer\_address**

|  |  |  |  |
| --- | --- | --- | --- |
| customer\_address | Type | NULL? | Table is used by **Queries**: |
| **ca\_address\_sk** | BIGINT | NOT NULL | Q7 Q9 Q17 |
| ca\_state | CHAR (2) |  | Q7 Q9 |
| ca\_country | VARCHAR (20) |  | Q9 |
| ca\_gmt\_offset | DECIMAL (5 ,2) |  | Q17 |

**customer\_demographics**

|  |  |  |  |
| --- | --- | --- | --- |
| customer\_  demographics | Type | NULL? | Table is used by **Queries**: |
| cd\_demo\_sk | BIGINT | NOT NULL | Q5 Q8 Q9 |
| cd\_gender | CHAR (1) |  | Q5 |
| cd\_marital\_status | CHAR (1) |  | Q9 |
| cd\_education\_status | CHAR (20) |  | Q5 Q9 |

**household\_demographics**

|  |  |  |  |
| --- | --- | --- | --- |
| household\_demographics | Type | NULL? | Table is used by **Queries**: |
| **hd\_demo\_sk** | BIGINT | NOT NULL | Q14 |
| hd\_dep\_count | INTEGER |  | Q14 |

**item**

|  |  |  |  |
| --- | --- | --- | --- |
| item | Type | NULL? | Table is used by **Queries**: |
| **i\_item\_sk** | BIGINT | NOT NULL | Q5 Q7 Q12 Q15 Q16 Q17 Q19 Q21 Q22 Q23 Q24 Q26 Q29 Q30 |
| i\_item\_id | CHAR (16) | NOT NULL | Q16 Q21 Q22 |
| i\_item\_desc | VARCHAR (200) |  | Q21 |
| i\_current\_price | DECIMAL (7 ,2) |  | Q7 Q22 Q24 |
| i\_class\_id | INTEGER |  | Q26 |
| i\_category\_id | INTEGER |  | Q1 Q15 Q29 Q30 |
| i\_category | CHAR (50) |  | Q5 Q7 Q12 Q17 Q26 |

**item\_marketprices**

|  |  |  |  |
| --- | --- | --- | --- |
| item\_marketprices | Type | NULL? | Table is used by **Queries**: |
| imp\_item\_sk | BIGINT | NOT NULL | Q24 |
| imp\_competitor\_ price | DECIMAL (7 ,2) |  | Q24 |
| imp\_start\_date | BIGINT |  | Q24 |
| imp\_end\_date | BIGINT |  | Q24 |

**inventory**

|  |  |  |  |
| --- | --- | --- | --- |
| inventory | Type | NULL? | Table is used by **Queries**: |
| **inv\_date\_sk** | BIGINT | NOT NULL | Q22 Q23 |
| inv\_item\_sk | BIGINT | NOT NULL | Q22 Q23 |
| inv\_warehouse\_sk | BIGINT | NOT NULL | Q22 Q23 |
| inv\_quantity\_on\_hand | INTEGER |  | Q22 Q23 |

**promotion**

|  |  |  |  |
| --- | --- | --- | --- |
| promotion | Type | NULL? | Table is used by **Queries**: |
| p\_channel\_dmail | CHAR (1) |  | Q17 |
| p\_channel\_email | CHAR (1) |  | Q17 |
| p\_channel\_tv, | CHAR (1) |  | Q17 |

**product\_reviews**

|  |  |  |  |
| --- | --- | --- | --- |
| product\_reviews | Type | NULL? | Table is used by **Queries**: |
| **pr\_review\_sk** | BIGINT | NOT NULL | Q27 Q28 |
| pr\_review\_date | DATE |  | Q18 |
| pr\_review\_rating | INT | NOT NULL | Q11 Q28 |
| pr\_item\_sk | BIGINT | NOT NULL | Q10 Q11 Q19 Q27 Q28 |
| pr\_review\_content | TEXT | NOT NULL | Q10 Q18 Q19 Q27 Q28 |

**store**

|  |  |  |  |
| --- | --- | --- | --- |
| store | Type | NULL? | Table is used by **Queries**: |
| **s\_store\_sk** | BIGINT | NOT NULL | Q9 Q17 Q18 Q21 |
| s\_store\_id | CHAR (16) | NOT NULL | Q21 |
| s\_store\_name | VARCHAR (50) |  | Q18 Q21 |
| s\_gmt\_offset | DECIMAL (5 ,2) |  | Q17 |

**store\_sales**

|  |  |  |  |
| --- | --- | --- | --- |
| store\_sales | Type | NULL? | Table is used by **Queries**: |
| **ss\_sold\_date\_sk** | BIGINT default 9999999 , |  | Q6 Q7 Q9 Q12 Q13 Q15 Q17 Q18 Q20 Q21 Q24 Q25 |
| ss\_sold\_time\_sk | BIGINT |  | Q12 |
| ss\_item\_sk, | BIGINT | NOT NULL | Q1 Q7 Q12 Q15 Q17 Q20 Q21 Q24 Q26 |
| ss\_customer\_sk | BIGINT |  | Q6 Q7 Q12 Q13 Q17 Q20 Q21 Q25 Q26 |
| ss\_cdemo\_sk | BIGINT |  | Q9 |
| ss\_addr\_sk | BIGINT |  | Q9 |
| ss\_store\_sk | BIGINT |  | Q1 Q9 Q15 Q17 Q18 Q21 |
| ss\_promo\_sk | BIGINT |  | Q17 |
| ss\_ticket\_number | BIGINT | NOT NULL | Q1 Q20 Q21 Q25 |
| ss\_quantity | INTEGER |  | Q21 Q24 |
| ss\_sales\_price | DECIMAL (7 ,2) |  | Q9 |
| ss\_ext\_discount\_amt | DECIMAL (7 ,2) |  | Q6 |
| ss\_ext\_sales\_price | DECIMAL (7 ,2) |  | Q6 Q17 |
| ss\_ext\_wholesale\_cost | DECIMAL (7 ,2) |  | Q6 |
| ss\_ext\_list\_price | DECIMAL (7 ,2) |  | Q6 |
| ss\_net\_paid | DECIMAL (7 ,2) |  | Q13 Q15 Q17 Q20 Q25 |
| ss\_net\_profit | DECIMAL (7 ,2) |  | Q9 |

**store\_returns**

|  |  |  |  |
| --- | --- | --- | --- |
| store\_returns | Type | NULL? | Table is used by **Queries**: |
| **sr\_returned\_date\_sk** | BIGINT default 9999999 |  | Q19 Q20 Q21 |
| sr\_item\_sk, | BIGINT | NOT NULL | Q19 Q20 Q21 |
| sr\_customer\_sk, | BIGINT |  | Q20 Q21 |
| sr\_ticket\_number | BIGINT | NOT NULL | Q20 Q21 |
| sr\_return\_quantity, | INTEGER |  | Q19 Q21 |
| sr\_return\_amt | DECIMAL (7 ,2) |  | Q20 |

**web\_sales**

|  |  |  |  |
| --- | --- | --- | --- |
| web\_sales | Type | NULL? | Table is used by **Queries**: |
| **ws\_sk** | BIGINT | NOT NULL | Q8 |
| ws\_sold\_date\_sk | BIGINT default 9999999 , |  | Q6 Q8 Q11 Q13 Q16 Q21 Q24 Q25 |
| ws\_sold\_time\_sk | BIGINT |  | Q14 |
| ws\_item\_sk | BIGINT | NOT NULL | Q11 Q16 Q21 Q24 Q29 |
| ws\_bill\_customer\_sk | BIGINT |  | Q6 Q13 Q21 Q25 Q29 |
| ws\_ship\_hdemo\_sk | BIGINT |  | Q14 |
| ws\_web\_page\_sk | BIGINT |  | Q14 |
| ws\_warehouse\_sk | BIGINT |  | Q16 Q22 |
| ws\_order\_number | BIGINT | NOT NULL | Q16 Q25 |
| ws\_quantit | INTEGER |  | Q21 Q24 |
| ws\_sales\_price | DECIMAL (7 ,2) |  | Q16 |
| ws\_ext\_discount\_amt | DECIMAL (7 ,2) |  | Q6 |
| ws\_ext\_sales\_price | DECIMAL (7 ,2) |  | Q6 |
| ws\_ext\_wholesale\_cost | DECIMAL (7 ,2) |  | Q6 |
| ws\_ext\_list\_price | DECIMAL (7 ,2) |  | Q6 |
| ws\_net\_paid | DECIMAL (7 ,2) |  | Q8 Q11 Q13 Q25 |

**web\_returns**

|  |  |  |  |
| --- | --- | --- | --- |
| web\_returns | Type | NULL? | Table is used by **Queries**: |
| wr\_returned\_date\_sk | BIGINT default 9999999 |  | Q19 |
| wr\_item\_sk | BIGINT | NOT NULL | Q16 Q19 |
| wr\_order\_number | BIGINT |  | Q16 |
| wr\_return\_quantity | INTEGER |  | Q19 |
| wr\_refunded\_cash | DECIMAL (7 ,2) |  | Q16 |

**web\_clickstreams**

|  |  |  |  |
| --- | --- | --- | --- |
| web\_clickstreams | Type | NULL? | Table is used by **Queries**: |
| **wcs\_click\_sk** | BIGINT | NOT NULL |  |
| wcs\_click\_date\_sk | BIGINT |  | Q3 Q4 Q8 Q12 |
| wcs\_click\_time\_sk | BIGINT |  | Q3 Q4 Q8 Q12 |
| wcs\_sales\_sk | BIGINT |  | Q3 Q8 |
| wcs\_item\_sk | BIGINT | can be null | Q2 Q3 Q4 Q5 Q8 Q12 Q30 |
| wcs\_web\_page\_sk | BIGINT |  | Q8 |
| wcs\_user\_sk | BIGINT | can be null | Q2 Q3 Q4 Q5 Q8 Q12 Q30 |

**warehouse**

|  |  |  |  |
| --- | --- | --- | --- |
| warehouse | Type | NULL? | Table is used by **Queries**: |
| w\_warehouse\_sk | BIGINT | NOT NULL | Q16 Q23 |
| w\_warehouse\_name | VARCHAR (20) |  | Q22 Q23 |
| w\_state | CHAR (2) |  | Q16 |

**web\_page**

|  |  |  |  |
| --- | --- | --- | --- |
| web\_page | Type | NULL? | Table is used by **Queries**: |
| **wp\_web\_page\_sk** | BIGINT | NOT NULL | Q4 Q8 Q14 |
| wp\_type | CHAR (50) |  | Q4 Q8 |
| wp\_char\_count | INTEGER |  | Q14 |

**web\_site**

(UNUSED/UNREFERENCED) only ref: web\_sales

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| web\_site |  | Type | NULL? | Table is used by **Queries**: |

**reason**

only referenced by sr\_reason\_sk and wr\_reason\_sk (both not used in **Queries**)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| reason |  | Type | NULL? | Table is used by **Queries**: |

**ship\_mode**

(UNUSED/UNREFERENCED)

|  |  |  |  |
| --- | --- | --- | --- |
| ship\_mode | Type | NULL? | Table is used by **Queries**: |

**income\_band**

( NOT USED)

|  |  |  |  |
| --- | --- | --- | --- |
| income\_band | Type | NULL? | Table is used by **Queries**: |

* + 1. Variables

|  |  |  |
| --- | --- | --- |
| **Global parameters (affect multiple tables)** |  |  |
| NULL\_CHANCE | 0.00025 | If a column is not “NOT NULL” some of the values may be null with the specified percentage |
| serial keys like customer\_sk date\_sk etc.. start at 0 or at 1 or... |  |  |
| ${SK\_ID\_OFFSET} | 0 | Serial key id offset.  Determines where serial keys of tables start |
|  |  |  |
| datetime format: yyyy-MM-dd HH:mm:ss |  |  |
| ${date\_begin\_date} | 01.01.1900 00:00 |  |
| ${date\_end\_date} | 01.01.2200 00:00 |  |
| ${CURRENT\_DAY} | 03.08.2005 00:00 | TODAY |
|  |  |  |
| ${one\_day\_in\_milliseconds} | 24.0 \* 60.0 \* 60.0 \* 1000.0 | one day |
|  |  |  |
| ${avg\_competitors\_per\_item} | 5 |  |
|  |  |  |
| ${anonymous\_reviews\_per\_item} | 5 |  |
| ${reviews\_per\_user} | 0.2 |  |
| ${reviews\_per\_sale} | 0.01 |  |
|  |  |  |
| ${pages\_per\_item} | 4 |  |
| ${pages\_to\_buy} | 4 |  |
| ${items\_per\_cart} | 12 |  |
| ${buy\_ratio} | 4 |  |
|  |  |  |
| **address** |  | Used in many tables like store\_sales, web\_sales, warehouse, etc.. |
| ${address\_street\_number\_min} | 1.0 |  |
| ${address\_street\_number\_max} | 1000.0 |  |
| ${address\_suite\_number\_min} | 1.0 |  |
| ${address\_suite\_number\_max} | 10.0 |  |
| ${address\_zip\_min} | 10000.0 |  |
| ${address\_zip\_max} | 99999.0 |  |
|  |  |  |
|  |  |  |
| **Table scaling types** |  |  |
| ${SF} | 1 | root" scalefactor. SF 1 ~1GB SF 10 ~10GB |
| ${SF\_log\_1.5} | Math.log(${SF}) / Math.log(1.5d) + 1.0d |  |
| ${SF\_log\_5} | (Math.log(${SF}) / Math.log(5.0d) + 1.0d) |  |
| ${SF\_sqrt} | Math.sqrt(${SF}) |  |
| ${SF\_linear} | ${SF} \* (2.0d - (${SF\_log\_5} \* ${SF\_sqrt} / ${SF})) |  |
|  |  |  |
|  |  |  |
| **Table specific properties** |  |  |
|  |  |  |
| **customer** |  |  |
| ${preferred\_cust\_likelihood} | 0.5 |  |
|  |  |  |
| **customer\_demographics** |  |  |
| ${gender\_likelihood} | 0.5 |  |
| ${married\_likelihood} | 0.3 |  |
| ${divorced\_likelihood} | 0.2 |  |
| ${single\_likelihood} | 0.2 |  |
| ${widowed\_likelihood} | 0.2 |  |
|  |  |  |
| **income\_band** |  |  |
| ${income\_band\_stepsize} | 10000 |  |
|  |  |  |
| **inventory** |  |  |
| ${inventory\_begin\_date} | 01.01.2001 00:00 |  |
| ${inventory\_end\_date} | 02.01.2006 00:00 |  |
| ${inventory\_weeks} | (${inventory\_end\_date} - ${inventory\_begin\_date}) / ${one\_day\_in\_milliseconds} / 7 |  |
| ${inventory\_days\_since\_date\_begin\_date} | (${inventory\_begin\_date} - ${date\_begin\_date}) / ${one\_day\_in\_milliseconds} |  |
|  |  |  |
| **item** |  |  |
| ${item\_begin\_date} | 03.01.2000 00:00 |  |
| ${item\_end\_date} | 05.01.2004 00:00 |  |
|  |  |  |
| **promotion** |  |  |
| ${dmail\_likelihood} | 0.5 |  |
| ${email\_likelihood} | 0.1 |  |
| ${catalog\_likelihood} | 0.1 |  |
| ${tv\_likelihood} | 0.1 |  |
| ${radio\_likelihood} | 0.1 |  |
| ${press\_likelihood} | 0.1 |  |
| ${event\_likelihood} | 0.1 |  |
| ${demo\_likelihood} | 0.1 |  |
| ${discount\_active\_likelihood} | 0.1 |  |
|  |  |  |
| **store** |  |  |
| ${store\_begin\_date} | 03.01.2000 00:00 |  |
| ${store\_end\_date} | 05.01.2004 00:00 |  |
| ${STORE\_MIN\_TAX\_PERCENTAGE} | 0.00 |  |
| ${STORE\_MAX\_TAX\_PERCENTAGE} | 0.11 |  |
|  |  |  |
| **store\_returns** |  |  |
| ${return\_store\_sale\_likelihood} | 0.1 |  |
|  |  |  |
| ${SR\_SAME\_CUSTOMER} | 0.8 |  |
|  |  |  |
| **store\_sales** |  |  |
| ${SS\_QUANTITY\_MAX} | 100 |  |
| ${SS\_WHOLESALE\_MAX} | 100.00 |  |
| ${SS\_MARKUP\_MAX} | 1.00 |  |
| ${SS\_DISCOUNT\_MAX} | 1.00 |  |
| ${store\_sales\_begin\_date} | 01.01.2001 00:00 |  |
| ${store\_sales\_end\_date} | 02.01.2006 00:00 |  |
| ${store\_sales\_days\_since\_date\_begin\_date} | (${store\_sales\_begin\_date} - ${date\_begin\_date}) / ${one\_day\_in\_milliseconds} |  |
| ${store\_sales\_days\_within} | (${store\_sales\_end\_date} - ${store\_sales\_begin\_date}) / ${one\_day\_in\_milliseconds} |  |
| ${SS\_ITEMS\_PER\_ORDER\_MIN} | 1 |  |
| ${SS\_ITEMS\_PER\_ORDER\_MAX} | 14 |  |
|  |  |  |
| **web\_clickstreams** |  |  |
| ${visitor\_likelihood} | 0.8 |  |
| ${visitor\_known\_likelihood} | 0.5 |  |
| ${mean\_clicks\_per\_visitor} | 16 |  |
| ${mean\_clicks\_per\_buyer} | 4 |  |
| ${clickstreams\_chunksize} | 5 |  |
|  |  |  |
| **web\_page** |  |  |
| ${web\_page\_begin\_date} | 03.01.2000 00:00 |  |
| ${web\_page\_end\_date} | 05.01.2004 00:00 |  |
| ${WP\_AUTOGEN\_PCT} | 0.30 |  |
| ${WP\_LINK\_MIN} | 2 |  |
| ${WP\_LINK\_MAX} | 25 |  |
| ${WP\_IMAGE\_MIN} | 1 |  |
| ${WP\_IMAGE\_MAX} | 7 |  |
| ${WP\_AD\_MIN} | 0 |  |
| ${WP\_AD\_MAX} | 4 |  |
|  |  |  |
| **warehouse** |  |  |
| ${W\_SQFT\_MIN} | 50000 |  |
| ${W\_SQFT\_MAX} | 1000000 |  |
|  |  |  |
| **web\_returns** |  |  |
| ${return\_web\_sale\_likelihood} | 0.1 |  |
|  |  |  |
| **web\_sales** |  |  |
| ${WS\_QUANTITY\_MAX} | 100 |  |
| ${WS\_WHOLESALE\_MAX} | 100.00 |  |
| ${WS\_MARKUP\_MAX} | 2.00 |  |
| ${WS\_DISCOUNT\_MAX} | 1.00 |  |
| ${WS\_MIN\_SHIP\_DELAY} | 1 |  |
| ${WS\_MAX\_SHIP\_DELAY} | 120 |  |
| ${WS\_ITEMS\_PER\_ORDER\_MIN} | 1 |  |
| ${WS\_ITEMS\_PER\_ORDER\_MAX} | 14 |  |
| ${WS\_GIFT\_PCT} | 0.07 |  |
| ${web\_sales\_begin\_date} | 01.01.2001 00:00 |  |
| ${web\_sales\_end\_date} | 02.01.2006 00:00 |  |
| ${web\_sales\_days\_since\_date\_begin\_date} | (${web\_sales\_begin\_date} - ${date\_begin\_date}) / ${one\_day\_in\_milliseconds} |  |
| ${web\_sales\_days\_within} | (${web\_sales\_end\_date} - ${web\_sales\_begin\_date}) / ${one\_day\_in\_milliseconds} |  |
|  |  |  |
| **table sizes formulas (Scaling of tables with increasing scale factor SF)** |  |  |
| **Not used tables:** |  |  |
| ${income\_band\_size} | 20 |  |
| ${reason\_size} | 35 \* ${SF\_log\_1.5} |  |
| ${ship\_mode\_size} | 20 |  |
| ${web\_site\_size} | 30 |  |
|  |  |  |
| **static (fixed size) tables** |  |  |
| ${date\_dim\_size} | (${date\_end\_date} - ${date\_begin\_date}) / ${one\_day\_in\_milliseconds} |  |
| ${time\_dim\_size} | ${one\_day\_in\_milliseconds} / 1000 |  |
| ${customer\_demographics\_size} | 1920800 |  |
| ${household\_demographics\_size} | 7200 |  |
|  |  |  |
| **normal not refreshed tables** |  |  |
| ${store\_size} | 12 \* ${SF\_sqrt} |  |
| ${promotion\_size} | 300 \* ${SF\_log\_1.5} |  |
| ${warehouse\_size} | 5.0d \* ${SF\_log\_5} |  |
| ${web\_page\_size} | 60 \* ${SF\_log\_1.5} |  |

* 1. Table Data Generation Rules

**date\_dim**

${date\_dim\_size} = 73049 (fixed, does not scale) one row per day in range:

(${date\_end\_date} - ${date\_begin\_date}) / ${one\_day\_in\_milliseconds})

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| date\_dim | Type | NULL? | Table is used by **Queries**: | Description  From: ${date\_begin\_date to: ${date\_end\_date} |
| **d\_date\_sk** | BIGINT | NOT NULL | Q4 Q6 Q7 Q9 Q13 Q16 Q17 Q19 Q21 Q22 Q23 | Key starting at 1 |
| d\_date\_id | CHAR (16) | NOT NULL |  | Unique String, len: 16  charset: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”  Example: AAAAAAAAOKJNECAA |
| d\_date | DATE |  | Q4 Q16 Q19 Q22 | From: ${date\_begin\_date to: ${date\_end\_date}  Format: yyyy-MM-dd |
| d\_month\_seq | INTEGER |  | Q7 | Starts at 0  Counts every month from start- to end-date |
| d\_week\_seq | INTEGER |  | Q19 | Dense unique sequence. Starts at ${SK\_ID\_OFFSET}  Counts every week from start- to end-date |
| d\_quarter\_seq | INTEGER |  |  | Starts at ${SK\_ID\_OFFSET}  Counts every quater from start- to end-date |
| d\_year | INTEGER |  | Q6 Q7 Q9 Q13 Q17 Q21 Q23 | Year Part of d\_date: yyyy |
| d\_dow | INTEGER |  |  | Day of week: 1-7, 1==Monday |
| d\_moy | INTEGER |  | Q7 Q17 Q21 Q23 | Month of year: 1-12, 1==Januar |
| d\_dom | INTEGER |  |  | Day of Month 1-31 |
| d\_qoy | INTEGER |  |  | Quarter of Year 1-4 |
| d\_fy\_year | INTEGER |  |  | Financial: d\_year + ½ year |
| d\_fy\_quarter\_seq | INTEGER |  |  | Financial: d\_quarter + ½ year |
| d\_fy\_week\_seq | INTEGER |  |  | Financial: d\_week + ½ year |
| d\_day\_name | CHAR (9) |  |  | Day of week d\_dow as string {Monday,…,Sunday} |
| d\_quarter\_name | CHAR (6) |  |  | Quarter of year d\_qoy as string yyyyQ{1..4}: example: 1990Q2 |
| d\_holiday | CHAR (1) |  |  | N/Y (true/false) |
| d\_weekend | CHAR (1) |  |  | N/Y (true/false) |
| d\_following\_holiday | CHAR (1) |  |  | N/Y (true/false) |
| d\_first\_dom | INTEGER |  |  | First day of month in Julian calendar (Julian day number e.g: 2415021) |
| d\_last\_dom | INTEGER |  |  | Last day of month in Julian calendar (Julian day number e.g: 2415021) |
| d\_same\_day\_ly | INTEGER |  |  | Same day in Julian calendar (Julian day number e.g: 2415021) |
| d\_same\_day\_lq | INTEGER |  |  | Same day in Julian calendar (Julian day number e.g: 2415021) |
| d\_current\_day | CHAR (1) |  |  | N/Y (true/false) d\_date\_sk==CURRENT\_DAY ? Y:N; |
| d\_current\_week | CHAR (1) |  |  | N/Y (true/false) d\_week\_seq==CURRENT\_WEEK ? Y:N; |
| d\_current\_month | CHAR (1) |  |  | N/Y (true/false) d\_moy==CURRENT\_MONTH ? Y:N; |
| d\_current\_quarter | CHAR (1) |  |  | N/Y (true/false) d\_qoy==CURRENT\_QUATER ? Y:N; |
| d\_current\_year | CHAR (1) |  |  | N/Y (true/false) d\_year==CURRENT\_YEAR ? Y:N; |
| Notes: |  |  |  |  |
| DISTRIBUTE BY REPLICATION ; |  |  |  |  |

**time\_dim**

${time\_dim\_size} = 86400 (fixed, does not scale) one row for every second in one day

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **time\_dim** | Type | NULL? | Table is used by **Queries**: | Description  Example:  0|AAAAAAAABAAAAAAA|0|0|0|0|AM|third|night|| |
| **t\_time\_sk** | BIGINT | NOT NULL | Q4 Q14 | Dense unique sequence. Starts at ${SK\_ID\_OFFSET} |
| t\_time\_id | CHAR (16) | NOT NULL |  | Unique String, len: 16  charset: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”  Example: AAAAAAAAOKJNECAA |
| t\_time | INTEGER |  | Q4 | Starts at 0  time\_id == t\_time\_sk |
| t\_hour | INTEGER |  | Q14 | time\_id/60/60 modulo 24 |
| t\_minute | INTEGER |  |  | time\_id/60 modulo 60 |
| t\_second | INTEGER |  |  | time\_id modulo 60 |
| t\_am\_pm | CHAR (2) |  |  | See Weighted list "purchase\_band" value col:1 weightColumn: 0 |
| t\_shift | CHAR (20) |  |  | See Weighted list "purchase\_band" value col:2 weightColumn: 0 |
| t\_sub\_shift | CHAR (20) |  |  | See Weighted list "purchase\_band" value col:3 weightColumn: 0 |
| t\_meal\_time | CHAR (20) |  |  | See Weighted list "purchase\_band" value col:4 weightColumn: 0 |
| Notes: |  |  |  |  |
| DISTRIBUTE BY REPLICATION ; |  |  |  |  |

**customer**

${customer\_size} = 100000 \* ${SF\_sqrt

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **customer** | Type | NULL? | Table is used by **Queries**: | Description |
| **c\_customer\_sk** | BIGINT | NOT NULL | Q5 Q6 Q7 Q13 Q17 | Dense unique sequence. Starts at ${SK\_ID\_OFFSET} |
| c\_customer\_id | CHAR (16) | NOT NULL | Q6 Q13 | Unique String, len: 16  charset: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”  Example: AAAAAAAAOKJNECAA |
| c\_current\_cdemo\_sk | BIGINT |  | Q5 | Random reference to table: customer\_demographics cd\_demo\_sk |
| c\_current\_hdemo\_sk | BIGINT |  |  | Random reference to table: household\_demographics hd\_demo\_sk |
| c\_current\_addr\_sk | BIGINT |  | Q7 Q17 | Random reference to table: customer\_address ca\_address\_sk |
| c\_first\_shipto\_date\_sk | BIGINT |  |  | Random reference to table: date\_dim d\_date\_sk |
| c\_first\_sales\_date\_sk | BIGINT |  |  | Random reference to table: date\_dim d\_date\_sk |
| c\_salutation | CHAR (10) |  |  | See Weighted list "salutations" value col:0 weightColumn: 0 or 1  Salution must match gender as implicitly choosen by: c\_first\_name |
| c\_first\_name | CHAR (20) |  | Q6 Q13 | See Weighted list "first\_names" value col:0 weightColumn: 0 |
| c\_last\_name | CHAR (30) |  | Q6 Q13 | See Weighted list "last\_names" value col:0 weightColumn: 0 |
| c\_preferred\_cust\_flag | CHAR (1) |  | Q6 | Propability : value  ${preferred\_cust\_likelihood}: Y  1-${preferred\_cust\_likelihood}: N |
| c\_birth\_day | INTEGER |  |  | Random number [1, 31] |
| c\_birth\_month | INTEGER |  |  | Random number [1, 12] |
| c\_birth\_year | INTEGER |  |  | Random number: [1924, 1992] |
| c\_birth\_country | VARCHAR (20) |  | Q6 | See Weighted list "countries" value col:0 weightColumn: 0 |
| c\_login | CHAR (13) |  | Q6 | Random string len: [1-13] |
| c\_email\_address | CHAR (50) |  | Q6 | Pattern: C\_first\_name.c\_last\_name@randomProvider.tld |
| c\_last\_review\_date | CHAR (10) |  |  | Min: ${CURRENT\_DAY} – 1 Year  Max: ${CURRENT\_DAY} |
| Note: |  |  |  |  |
| DISTRIBUTE BY HASH ( c\_customer\_sk ); |  |  |  |  |

**customer\_address**

${customer\_address\_size} = ${customer\_size} / 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **customer\_address** | Type | NULL? | Table is used by **Queries**: | Description |
| **ca\_address\_sk** | BIGINT | NOT NULL | Q7 Q9 Q17 | Dense unique sequence. Starts at ${SK\_ID\_OFFSET} |
| ca\_address\_id | CHAR (16) | NOT NULL |  | Unique String, len: 16  charset: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”  Example: AAAAAAAAOKJNECAA |
| ca\_street\_number | CHAR (10) |  |  | Random number: [ 1, 1000] |
| ca\_street\_name | VARCHAR (60) |  |  | Probability:  50% 1 word “%s”  50% 2 Words “%s %s”  From Weighted list "street\_names", valueCol:0 weightCoL 0 |
| ca\_street\_type | CHAR (15) |  |  | See Weighted list "street\_type" value col:0 weightColumn: 0 |
| ca\_suite\_number | CHAR (10) |  |  | Random String len: [1, 10] |
| ca\_city | VARCHAR (60) |  |  | See Weighted list "cities" value col:0 weightColumn: 0 |
| ca\_county | VARCHAR (30) |  |  | See Weighted list "fips\_county" value col:county: weightColumn: uniform |
| ca\_state | CHAR (2) |  | Q7 Q9 | See Weighted list "fips\_county" value col: st weightColumn: uniform  Same entry as ca\_county (state must match county) |
| ca\_zip | CHAR (10) |  |  | Random number [10000, 99999] |
| ca\_country | VARCHAR (20) |  | Q9 | 'United States' |
| ca\_gmt\_offset | DECIMAL (5 ,2) |  | Q17 | See Weighted list "fips\_county" value col:gmt weightColumn: uniform  Same entry as ca\_county (state must match county) |
| ca\_location\_type | CHAR (20) |  |  | See Weighted list "location\_type" value col:0 weightColumn: 0 |
| Note: |  |  |  |  |
| DISTRIBUTE BY HASH ( ca\_address\_sk ); |  |  |  |  |

**customer\_demographics**

${customer\_demographics\_size} = 1920800 (fixed, does not scale)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **customer\_**  **demographics** | Type | NULL? | Table is used by **Queries**: | Description |
| cd\_demo\_sk | BIGINT | NOT NULL | Q5 Q8 Q9 | Dense unique sequence. Starts at ${SK\_ID\_OFFSET} |
| cd\_gender | CHAR (1) |  | Q5 | See Weighted list "gender" value col:0 weightColumn: 0 |
| cd\_marital\_status | CHAR (1) |  | Q9 | See Weighted list "marital\_status" value col:0 weightColumn: 0 |
| cd\_education\_status | CHAR (20) |  | Q5 Q9 | See Weighted list " education" value col:0 weightColumn: 0 |
| cd\_purchase\_estimate | INTEGER |  |  | See Weighted list "purchase\_band" value col:0 weightColumn: 0 |
| cd\_credit\_rating | CHAR (10) |  |  | See Weighted list "credit\_rating" value col:0 weightColumn: 0 |
| cd\_dep\_count | INTEGER |  |  | See Weighted list "dependent\_count" value col:0 weightColumn: 0 |
| cd\_dep\_employed\_count | INTEGER |  |  | See Weighted list "dependent\_count" value col:0 weightColumn: 0 |
| cd\_dep\_college\_count | INTEGER |  |  | See Weighted list "dependent\_count" value col:0 weightColumn: 0 |
| Note: |  |  |  |  |
| DISTRIBUTE BY REPLICATION ; |  |  |  |  |

**household\_demographics**

${household\_demographics\_size} = 7200 (fixed, does not scale)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **household\_demographics** | Type | NULL? | Table is used by **Queries**: | Description |
| **hd\_demo\_sk** | BIGINT | NOT NULL | Q14 | Dense unique sequence. Starts at ${SK\_ID\_OFFSET}  referenced by:   * ws\_ship\_hdemo\_sk * c\_current\_hdemo\_sk * ss\_hdemo\_sk * sr\_hdemo\_sk * ws\_bill\_hdemo\_sk * ws\_ship\_hdemo\_sk * wr\_refunded\_hdemo\_sk * wr\_returning\_hdemo\_sk |
| hd\_income\_band\_sk | BIGINT |  |  | Random reference to table: “income\_band” ib\_income\_band\_sk |
| hd\_buy\_potential | CHAR (15) |  |  | See Weighted list "buy\_potentialt" value col:0 weightColumn: 0 |
| hd\_dep\_count | INTEGER |  | Q14 | See Weighted list "dependent\_count" value col:0 weightColumn: 0 |
| hd\_vehicle\_count | INTEGER |  |  | See Weighted list "vehicle\_count" value col:0 weightColumn: 0 |
| Notes: |  |  |  |  |
| DISTRIBUTE BY REPLICATION ; |  |  |  |  |

**item**

${item\_size} = 18000.0 \* ${SF\_sqrt}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **item** | Type | NULL? | Table is used by **Queries**: | Description |
| **i\_item\_sk** | BIGINT | NOT NULL | Q5 Q7 Q12 Q15 Q16 Q17 Q19 Q21 Q22 Q23 Q24 Q26 Q29 Q30 | Dense unique sequence. Starts at ${SK\_ID\_OFFSET} |
| i\_item\_id | CHAR (16) | NOT NULL | Q16 Q21 Q22 | Unique String, len: 16  charset: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”  Example: AAAAAAAAOKJNECAA |
| i\_rec\_start\_date | DATE |  |  | Date  from: ${item\_begin\_date}  to: ${inventory\_begin\_date}  Format: yyyy-MM-dd  With: i\_rec\_start\_date{n} < i\_rec\_start\_date{n+1}  where n= **i\_item\_sk** |
| i\_rec\_end\_date, | DATE |  |  | No end date for the moment. Value: “”  Elese:  50% Empty 50%: i\_rec\_start\_date + rand[2years, 4years] |
| i\_item\_desc | VARCHAR (200) |  | Q21 | Sentences following pseudo englisch gramatic  Example:  Clear circumstances know then further white companies. Typical budgets take both required children. Appeals must not make civil, financial representatives. Emotional areas shall wear only. |
| i\_current\_price | DECIMAL (7 ,2) |  | Q7 Q22 Q24 | Random decimal [0.09, 99.99] |
| i\_wholesale\_cost | DECIMAL (7 ,2) |  |  | Random decimal [0.02, 87,36] |
| i\_brand\_id | INTEGER |  |  | Radnom integer [1001001, 10016017] |
| i\_brand | CHAR (50) |  |  | Random string len [1, 50] |
| i\_class\_id | INTEGER |  | Q26 | Unique ID identifying i\_class. starts at a |
| i\_class | CHAR (50) |  |  | Class must depend on selected i\_category !  See the following WeightedLists mathing the selected i\_category:  Women -> women\_class  Men -> men\_class  Children -> children\_class  Shoes -> shoe\_class  Music -> music\_class  Jewelry -> jewelry\_class  Home -> home\_class  Sports -> sport\_class  Books -> book\_class  Electronics -> electronic\_class |
| i\_category\_id | INTEGER |  | Q1 Q15 Q29 Q30 | Unique id identifying i\_category. Starts at 1 |
| i\_category | CHAR (50) |  | Q5 Q7 Q12 Q17 Q26 | See Weighted list "categories" value col:0 weightColumn: 0 |
| i\_manufact\_id | CHAR (50) |  |  | Random integer [1, 1000] |
| i\_manufact | CHAR (50) |  |  | Random String len [1, 50] |
| i\_size | CHAR (20) |  |  | See Weighted list "sizes" value col:0 weightColumn: 0 |
| i\_formulation, | CHAR (20) |  |  | Random String len [1, 20] |
| i\_color | CHAR (20) |  |  | See Weighted list "color" value col:0 weightColumn: 0 |
| i\_units | CHAR (10) |  |  | See Weighted list "units" value col:0 weightColumn: 0 |
| i\_container | CHAR (10) |  |  | See Weighted list "container" value col:0 weightColumn: 0 |
| i\_manager\_id | INTEGER |  |  | Random integer [1, 1000] distibuted like:  Weighted list "i\_manager\_id" |
| i\_product\_name | CHAR (50) |  |  | Random String len [1, 50] |
|  |  |  |  |  |
| Notes: |  |  |  |  |
| DISTRIBUTE BY HASH ( i\_item\_sk ); |  |  |  |  |

**item\_marketprices**

${item\_marketprices\_size} = ${item\_size} \* ${avg\_competitors\_per\_item}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **item\_marketprices** | Type | NULL? | Table is used by **Queries**: | Description |
| **imp\_sk** | BIGINT | NOT NULL |  | Dense unique sequence. Starts at ${SK\_ID\_OFFSET} |
| imp\_item\_sk | BIGINT | NOT NULL | Q24 | Random reference to table:: item i\_item\_sk |
| imp\_competitor | VARCHAR (20) |  |  | Random String len [1, 20] |
| imp\_competitor\_ price | DECIMAL (7 ,2) |  | Q24 | Random decimal [0.09, 99.99] |
| imp\_start\_date | BIGINT |  | Q24 | Random reference to table: date d\_date\_sk |
| imp\_end\_date | BIGINT |  | Q24 | Random reference to table: date d\_date\_sk > imp\_start\_date |
| Notes: |  |  |  |  |
| DISTRIBUTE BY HASH ( imp\_sk ); |  |  |  |  |

**inventory**

${inventory\_size}= ${inventory\_weeks} \* ${item\_size} \* ${warehouse\_size}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **inventory** | Type | NULL? | Table is used by **Queries**: | Description |
| **inv\_date\_sk** | BIGINT | NOT NULL | Q22 Q23 | (id or row) / ${item\_size} / ${warehouse\_size}) \* 7 + ${inventory\_days\_since\_date\_begin\_date} |
| inv\_item\_sk | BIGINT | NOT NULL | Q22 Q23 | (id or row) modulo ${item\_size} |
| inv\_warehouse\_sk | BIGINT | NOT NULL | Q22 Q23 | (id or row) / ${item\_size} modlulo ${warehouse\_size} |
| inv\_quantity\_on\_hand | INTEGER |  | Q22 Q23 | Random integer between [0, 1000] |
| Notes: |  |  |  |  |
| DISTRIBUTE BY HASH ( inv\_item\_sk ); |  |  |  |  |

**promotion**

${promotion\_size} = 300 \* ${SF\_log\_1.5}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **promotion** | Type | NULL? | Table is used by **Queries**: | Description |
| **p\_promo\_sk** | BIGINT | NOT NULL | Q17 | Dense unique sequence. Starts at ${SK\_ID\_OFFSET} |
| p\_promo\_id | CHAR (16) | NOT NULL |  | Unique String, len: 16  charset: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”  Example: AAAAAAAAOKJNECAA |
| p\_start\_date\_sk | BIGINT |  |  | Random reference to table: date d\_date\_sk |
| p\_end\_date\_sk | BIGINT |  |  | Random reference to table: date d\_date\_sk > p\_start\_date\_sk |
| p\_item\_sk | BIGINT |  |  | Random reference to table: item i\_item\_sk |
| p\_cost | DECIMAL (15 ,2) |  |  | Random decimal [10.00, 1000.00] |
| p\_response\_target | INTEGER |  |  | 1 |
| p\_promo\_name | CHAR (50) |  |  | See Weighted list "syllables" value col:0 weightColumn: 0 |
| p\_channel\_dmail | CHAR (1) |  | Q17 | Propability : value  ${dmail\_likelihood}: Y  1-${dmail\_likelihood}: N |
| p\_channel\_email | CHAR (1) |  | Q17 | Propability : value  ${email\_likelihood}: Y  1-${email\_likelihood}: N |
| p\_channel\_catalog | CHAR (1) |  |  | Propability : value  ${catalog\_likelihood}: Y  1-${catalog\_likelihood}: N |
| p\_channel\_tv, | CHAR (1) |  | Q17 | Propability : value  ${tv\_likelihood}: Y  1-${tv\_likelihood}: N |
| p\_channel\_radio | CHAR (1) |  |  | Propability : value  ${radio\_likelihood}: Y  1-${radio\_likelihood}: N |
| p\_channel\_press | CHAR (1) |  |  | Propability : value  ${press\_likelihood}: Y  1-${press\_likelihood}: N |
| p\_channel\_event | CHAR (1) |  |  | Propability : value  ${event\_likelihood}: Y  1-${event\_likelihood}: N |
| p\_channel\_demo | CHAR (1) |  |  | Propability : value  ${channel\_likelihood}: Y  1-${channel\_likelihood}: N |
| p\_channel\_details | VARCHAR (100) |  |  | Sentences following pseudo englisch gramatic  Example:  Clear circumstances know then further white companies. Typical budgets take both required children. Appeals must not make civil, financial representatives. Emotional areas shall wear only. |
| p\_purpose, | CHAR (15) |  |  | create promo\_purpose;  set types = (varchar);  set weights = 1;  add ("Unknown": 4); |
| p\_discount\_active | CHAR (1) |  |  | Propability : value  ${discount\_active\_likelihood}: Y  1-${discount\_active\_likelihood}: N |
| Note: |  |  |  |  |
| ) DISTRIBUTE BY REPLICATION ; |  |  |  |  |

**product\_reviews**

${product\_reviews\_size} = (${item\_size} \* ${anonymous\_reviews\_per\_item}) + (${web\_sales\_size} \* ${reviews\_per\_sale})

pr\_review\_content must contain sentences which match the refrenced item type and rating.

The benchmark contains many semantic analysis **Queries**, trying to classify the reviews based on the user written text. Therfore, pr\_review\_content must resemble a human written review text as close as possible!

If the referenced item is a DVD-Player with rating 5, a human reader should be able to recognize that the computer generated review is indeed talking about such a DVD-Player product and that the writer was satisfied. A rating of 1 should reflect a negative review.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **product\_reviews** | Type | NULL? | Table is used by **Queries**: | Description |
| **pr\_review\_sk** | BIGINT | NOT NULL | Q27 Q28 | Dense unique sequence. Starts at ${SK\_ID\_OFFSET} |
| pr\_review\_date | DATE |  | Q18 | Date  from: ${date\_begin\_date}  to: ${date\_begin\_date}  Format: yyyy-MM-dd  With: i\_rec\_start\_date{n} < i\_rec\_start\_date{n+1}  where n= **i\_item\_sk** |
| pr\_review\_time | CHAR(6) |  |  | Random reference to table: time\_dim t\_time\_sk |
| pr\_review\_rating | INT | NOT NULL | Q11 Q28 | 1-5, See Weighted list "ratingWeights" value col:0 weightColumn: 0 |
| pr\_item\_sk | BIGINT | NOT NULL | Q10 Q11 Q19 Q27 Q28 | Random reference to a ws\_item\_sk of referenced order\_sk in pr\_order\_sk |
| pr\_user\_sk | BIGINT |  |  | Random reference to ws\_user\_sk of referenced order\_sk in pr\_order\_sk |
| pr\_order\_sk | BIGINT |  |  | Random reference to web\_sales order\_id |
| pr\_review\_content | TEXT | NOT NULL | Q10 Q18 Q19 Q27 Q28 | pr\_review\_content must contain sentences which match the refrenced item’s type (i\_category) and pr\_review\_rating. |
| Notes: |  |  |  |  |
| DISTRIBUTE BY HASH ( pr\_review\_sk ); |  |  |  |  |

**store**

${store\_size} =12 \* ${SF\_sqrt}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **store** | Type | NULL? | Table is used by **Queries**: | Description  Example:  1|AAAAAAAAAAAAAAAA|1997-03-13||2451189|ought|245|5250760|8AM-4PM|William Ward|2|Unknown|Enough high areas stop expectations. Elaborate, local is|Charles Bartley|1|Unknown|1|Unknown|767|Spring |Wy|Suite 250|Midway|Williamson County|TN|31904|United States|-5|0.03| |
| **s\_store\_sk** | BIGINT | NOT NULL | Q9 Q17 Q18 Q21 | Dense unique sequence. Starts at ${SK\_ID\_OFFSET} |
| s\_store\_id | CHAR (16) | NOT NULL | Q21 | Unique String, len: 16  charset: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”  Example: AAAAAAAAOKJNECAA |
| s\_rec\_start\_date | DATE |  |  | Date  from: ${store\_begin\_date}  to: ${store\_begin\_date}  Format: yyyy-MM-dd  With: s\_rec\_start\_date{n} < s\_rec\_start\_date{n+1}  where n= **s\_store\_sk** |
| s\_rec\_end\_date | DATE |  |  | No end date for the moment. Value: “”  Elese:  50% Empty 50%: wp\_rec\_start\_date + rand[2years, 4years] |
| s\_closed\_date\_sk | BIGINT |  |  | With STORE\_CLOSED\_PCT probabiltity a store is closed.  If closed: ref to table date d\_date\_sk |
| s\_store\_name | VARCHAR (50) |  | Q18 Q21 | One random word from Weighted List ‘syllables’ |
| s\_number\_employees | INTEGER |  |  | Random integer between: [200, 300] |
| s\_floor\_space | INTEGER |  |  | Random integer between: [5000000, 10000000] |
| s\_hours | CHAR (20) |  |  | Weighted List ‘call\_center\_hours’, value\_col= 0; weight\_col: 0 |
| s\_manager | VARCHAR (40) |  |  | Pattern: "%s %s"  Weighted List ‘first\_names’, value\_col= 0; weight\_col: 0  Weighted List ‘last\_names’; value\_col= 0; weight\_col: 0 |
| s\_market\_id | INTEGER |  |  | Random integer between: [2, 10] |
| s\_geography\_class | VARCHAR (100) |  |  | Value: "Unknown" |
| s\_market\_desc | VARCHAR (100) |  |  | Sentences following pseudo englisch gramatic  Example:  Clear circumstances know then further white companies. Typical budgets take both required children. Appeals must not make civil, financial representatives. Emotional areas shall wear only. |
| s\_market\_manager | VARCHAR (40) |  |  | Pattern: "%s %s"  Weighted List ‘first\_names’, value\_col= 0; weight\_col: 0  Weighted List ‘last\_names’; , value\_col= 0; weight\_col: 0 |
| s\_division\_id | INTEGER |  |  | Value: 1 |
| s\_division\_name | VARCHAR (50) |  |  | Value: "Unknown" |
| s\_company\_id | INTEGER |  |  | Value: 1 |
| s\_company\_name | VARCHAR (50) |  |  | Value: "Unknown" |
| s\_street\_number | VARCHAR (10) |  |  | Address like in warehouse |
| s\_street\_name | VARCHAR (60) |  |  | Address like in warehouse |
| s\_street\_type | CHAR (15) |  |  | Address like in warehouse |
| s\_suite\_number | CHAR (10) |  |  | Address like in warehouse |
| s\_city | VARCHAR (60) |  |  | Address like in warehouse |
| s\_county | VARCHAR (30) |  |  | Address like in warehouse |
| s\_state | CHAR (2) |  |  | Address like in warehouse |
| s\_zip | CHAR (10) |  |  | Address like in warehouse |
| s\_country | VARCHAR (20) |  |  | Address like in warehouse |
| s\_gmt\_offset | DECIMAL (5 ,2) |  | Q17 | Address like in warehouse |
| s\_tax\_precentage | DECIMAL (5 ,2) |  |  | UNIFORM RAND DECIMAL between [${STORE\_MIN\_TAX\_PERCENTAGE}, ${STORE\_MAX\_TAX\_PERCENTAGE}] |
| Note: |  |  |  |  |
| DISTRIBUTE BY REPLICATION ; |  |  |  |  |

**store\_sales**

${store\_sales\_size} = 90000.0d \* ${SF\_linear}

One logical sale consists of random[${SS\_ITEMS\_PER\_ORDER\_MIN}, ${SS\_ITEMS\_PER\_ORDER\_MAX] itmes.

|  |  |
| --- | --- |
| **Logical sale** | N = random[${SS\_ITEMS\_PER\_ORDER\_MIN}, ${SS\_ITEMS\_PER\_ORDER\_MAX]  1=same for every N  Write N lines for a logical Sale into store\_sales table |
| **ss\_sold\_date\_sk** | 1 |
| ss\_sold\_time\_sk | 1 |
| ss\_item\_sk | N |
| ss\_customer\_sk | 1 |
| ss\_cdemo\_sk | 1 |
| ss\_hdemo\_sk | 1 |
| ss\_addr\_sk | 1 |
| ss\_store\_sk | 1 |
| ss\_promo\_sk | 1 |
| ss\_ticket\_number | 1 |
| ss\_quantity | N |
| ss\_wholesale\_cost | N |
| ss\_list\_price | N |
| ss\_sales\_price | N |
| ss\_ext\_discount\_amt | N |
| ss\_ext\_sales\_price | N |
| ss\_ext\_wholesale\_cost | N |
| ss\_ext\_list\_price | N |
| ss\_ext\_tax | N |
| ss\_coupon\_amt | N |
| ss\_net\_paid | N |
| ss\_net\_paid\_inc\_tax | N |
| ss\_net\_profit | N |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **store\_sales** | Type | NULL? | Table is used by **Queries**: | Description |
| **ss\_sold\_date\_sk** | BIGINT default 9999999 , |  | Q6 Q7 Q9 Q12 Q13 Q15 Q17 Q18 Q20 Q21 Q24 Q25 | ${store\_sales\_days\_since\_date\_begin\_date} + Math.floor( (current row or id) \* ((${store\_sales\_days\_within} - 1) / ${store\_sales\_size}))  Implicit references to date d\_date\_sk |
| ss\_sold\_time\_sk | BIGINT |  | Q12 | Random reference to table: time\_dim t\_time\_sk |
| ss\_item\_sk, | BIGINT | NOT NULL | Q1 Q7 Q12 Q15 Q17 Q20 Q21 Q24 Q26 | PrimaryKey;  Random reference to item i\_item\_sk  A logical sale (same ss\_ticket\_number) consist of  random[${SS\_ITEMS\_PER\_ORDER\_MIN}, ${SS\_ITEMS\_PER\_ORDER\_MAX] itmes. |
| ss\_customer\_sk | BIGINT |  | Q6 Q7 Q12 Q13 Q17 Q20 Q21 Q25 Q26 | Random reference to table: customer c\_customer\_sk |
| ss\_cdemo\_sk | BIGINT |  | Q9 | same cdemo\_sk as referenced customer selected in ss\_customer\_sk |
| ss\_hdemo\_sk | BIGINT |  |  | same hdemo\_sk as referenced customer selected in ss\_customer\_sk |
| ss\_addr\_sk | BIGINT |  | Q9 | same addr\_sk as referenced customer selected in ss\_customer\_sk |
| ss\_store\_sk | BIGINT |  | Q1 Q9 Q15 Q17 Q18 Q21 | Random reference to s\_store\_sk |
| ss\_promo\_sk | BIGINT |  | Q17 | Random reference to p\_promo\_sk |
| ss\_ticket\_number | BIGINT | NOT NULL | Q1 Q20 Q21 Q25 | Dense unique sequence. Starts at ${SK\_ID\_OFFSET} |
| ss\_quantity | INTEGER |  | Q21 Q24 | Purchased quantity of item  Random integer from [1, ${SS\_QUANTITY\_MAX}] |
| ss\_wholesale\_cost | DECIMAL (7 ,2) |  |  | Random decimal from [1, ${SS\_WHOLESALE\_MAX}] |
| ss\_list\_price | DECIMAL (7 ,2) |  |  | List price of single item:  ss\_wholesale\_cost \* (1 + random[0.00, ${SS\_MARKUP\_MAX} ] ) |
| ss\_sales\_price | DECIMAL (7 ,2) |  | Q9 | Sales price of single item:  ss\_listPrice \* (1 - random[0.00, ${SS\_DISCONUT\_MAX} ] ) |
| ss\_ext\_discount\_amt | DECIMAL (7 ,2) |  | Q6 | Discount of item times quantity:  ss\_ext\_list\_price - ss\_ext\_sales\_price |
| ss\_ext\_sales\_price | DECIMAL (7 ,2) |  | Q6 Q17 | Sales price of item times quantity:  ss\_sales\_price \* ss\_quantity |
| ss\_ext\_wholesale\_cost | DECIMAL (7 ,2) |  | Q6 | Wholesale cost of item times quantity  ss\_wholesale\_cost \* ss\_quantity |
| ss\_ext\_list\_price | DECIMAL (7 ,2) |  | Q6 | List price of item times quantity  ss\_list\_price \* ss\_quantity |
| ss\_ext\_tax | DECIMAL (7 ,2) |  |  | Random[0.00, 0.09] \* ss\_net\_paid |
| ss\_coupon\_amt | DECIMAL (7 ,2) |  |  | Coupon discount  Probability:  0.8: value: 0.00  0.2: Value: ss\_ext\_sales\_price \* random[0.00, 1.00] |
| ss\_net\_paid | DECIMAL (7 ,2) |  | Q13 Q15 Q17 Q20 Q25 | Net paid of item times quantity  ss\_ext\_sales\_price \* ss\_coupon\_amt |
| ss\_net\_paid\_inc\_tax | DECIMAL (7 ,2) |  |  | Net paid including tax of item times quantity  ss\_net\_paid +ss\_ext\_tax |
| ss\_net\_profit | DECIMAL (7 ,2) |  | Q9 | Profit on that item purchase  ss\_net\_paid - ss\_ext\_wholesale\_cost |
| Notes: |  |  |  |  |
| DISTRIBUTE BY HASH ( ss\_item\_sk ); |  |  |  |  |

**store\_returns**

${store\_returns\_size} = ${return\_store\_sale\_likelihood} \* ${store\_sales\_size}

Store\_returns contains returned items for store\_sales. A logical store\_sale is identified by ss\_ticket\_id. This table must not contain more the one logical return entry for the same ss\_ticket\_id.

If a store sale is returned, a customer may not return the complete order, but only some items from it. Additionally he may have purchased 10 units of a certain item, but only returns, e.g., 5 of them. Return not all but random 1-N items from a selected store\_sale. Like in store\_sales, one logical “store\_return” contains multiple items and produces a store\_sales table row per returned item.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ss\_sold\_date\_sk | ss\_item\_sk, | ss\_customer\_sk | ss\_ticket\_number | ss\_quantity | ss\_wholesale\_cost | ss\_list\_price | ss\_sales\_price | ss\_ext\_list\_price |
| 04.09.2004 | 23 | 2345 | 1 | 5 | 45,40 | 54,35 | 26,44 | 271,73 |
| 04.09.2004 | 76 | 2345 | 1 | 1 | 23,23 | 29,62 | 8,67 | 29,62 |
| 04.09.2004 | 365 | 2345 | 1 | 7 | 25,52 | 37,92 | 33,65 | 265,41 |
| 05.09.2004 | 637 | 734 | 2 | 1 | 65,52 | 92,03 | 26,69 | 276,08 |
| 05.09.2004 | 345 | 734 | 2 | 3 | 24,48 | 48,45 | 1,80 | 48,45 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| sr\_return\_date\_sk | sr\_item\_sk, | sr\_customer\_sk | sr\_ticket\_number | sr\_quantity | sr\_wholesale\_cost | sr\_list\_price | sr\_sales\_price | sr\_ext\_list\_price |
| 19.09.2004 | 23 | 2345 | 1 | 5 (of 5) | 45,40 | 54,35 | 26,44 | 271,73 |
| 19.09.2004 | 365 | 2345 | 1 | 1 (of 7) | 25,52 | 37,92 | 33,65 | 37,92 |
| 08.12.2008 | 637 | 734 | 2 | 2 (of 3) | 65,52 | 92,03 | 26,69 | 184,06 |
| 05.09.2004 | 345 | 734 | 2 | 3 | 24,48 | 48,45 | 1,80 | 48,45 |

|  |  |
| --- | --- |
| **Logical sale** | Pick a random unique store\_sale ticket\_number. The selected store\_sale consists of N items. From these N items return random M items.  M=random[1, N]  1=same for every M  Write M lines for a logical return into store\_returns table |
| **sr\_returned\_date\_sk** | **1** |
| sr\_return\_time\_sk, | 1 |
| sr\_item\_sk, | M |
| sr\_customer\_sk, | 1 |
| sr\_cdemo\_sk, | 1 |
| sr\_hdemo\_sk, | 1 |
| sr\_addr\_sk, | 1 |
| sr\_store\_sk, | 1 |
| sr\_reason\_sk, | N |
| sr\_ticket\_number | 1 |
| sr\_return\_quantity, | N |
| sr\_return\_amt | M |
| sr\_return\_tax | M |
| sr\_return\_amt\_inc\_tax | M |
| sr\_fee | M |
| sr\_return\_ship\_cost | M |
| sr\_refunded\_cash | M |
| sr\_reversed\_charge | M |
| sr\_store\_credit | M |
| sr\_net\_loss, | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **store\_returns** | Type | NULL? | Table is used by **Queries**: | Description |
| **sr\_returned\_date\_sk** | BIGINT default 9999999 |  | Q19 Q20 Q21 | Random reference to date after! referenced store\_sales ss\_soled\_date\_sk with same ticket number |
| sr\_return\_time\_sk, | BIGINT |  |  | Random reference to time\_dim t\_time\_sk |
| sr\_item\_sk, | BIGINT | NOT NULL | Q19 Q20 Q21 | Random [1-N] item\_sk’s from ss\_item\_sk’s in referenced store\_sales ss\_ticket\_number (not necessary only one or all items from a store\_sales ticket are returned) |
| sr\_customer\_sk, | BIGINT |  | Q20 Q21 | Reference to customer\_sk, same as in store\_sales with same ticket number |
| sr\_cdemo\_sk, | BIGINT |  |  | Reference to cdemo\_sk, same as in store\_sales with same ticket number |
| sr\_hdemo\_sk, | BIGINT |  |  | Reference to hdemo\_sk, same as in store\_sales with same ticket number |
| sr\_addr\_sk, | BIGINT |  |  | Reference to addr\_sk, same as in store\_sales with same ticket number |
| sr\_store\_sk, | BIGINT |  |  | Reference to store\_sk, same as in store\_sales with same ticket number |
| sr\_reason\_sk, | BIGINT |  |  | random Reference to reason r\_reason\_sk for every returned item |
| sr\_ticket\_number | BIGINT | NOT NULL | Q20 Q21 | Reference a uniqe existing ticket from store\_sales ss\_ticket\_number |
| sr\_return\_quantity, | INTEGER |  | Q19 Q21 | M, Number of returned items in this logical return. |
| sr\_return\_amt | DECIMAL (7 ,2) |  | Q20 | ss\_sales\_price \* sr\_return\_quantity |
| sr\_return\_tax | DECIMAL (7 ,2) |  |  | sr\_return\_amt \* tax\_pct  with tax\_pct = random decimal between [0.00, 0.09] |
| sr\_return\_amt\_inc\_tax | DECIMAL (7 ,2) |  |  | sr\_return\_amt + sr\_return\_tax |
| sr\_fee | DECIMAL (7 ,2) |  |  | Random decimal between [0.50, 100.00] |
| sr\_return\_ship\_cost | DECIMAL (7 ,2) |  |  | ss\_list\_price \* shipping(=randDecimal[0.00, 1.00] \* sr\_return\_quantity |
| sr\_refunded\_cash | DECIMAL (7 ,2) |  |  | rand[0.0,1.0] \* sr\_return\_amt |
| sr\_reversed\_charge | DECIMAL (7 ,2) |  |  | rand[0.01, 1.00] \* (sr\_return\_amt - sr\_refunded\_cash) |
| sr\_store\_credit | DECIMAL (7 ,2) |  |  | sr\_return\_amt - sr\_reversed\_charge - sr\_refunded\_cash |
| sr\_net\_loss, | DECIMAL (7 ,2) |  |  | sr\_net\_loss = sr\_return\_amt + sr\_return\_ship\_cost + sr\_return\_tax - sr\_store\_credit - sr\_refunded\_cash - sr\_reversed\_charge + sr\_fee |
| Notes: |  |  |  |  |
| DISTRIBUTE BY HASH ( sr\_item\_sk ); |  |  |  |  |

**web\_sales**

${web\_sales\_size} = 90000.0d \* ${SF\_linear}

One logical sale consists of random[${WS\_ITEMS\_PER\_ORDER\_MIN}, ${WS\_ITEMS\_PER\_ORDER\_MAX] itmes.

|  |  |
| --- | --- |
| **Logical sale** | N = random[${WS\_ITEMS\_PER\_ORDER\_MIN}, ${WS\_ITEMS\_PER\_ORDER\_MAX]  1=same for every N  Write N lines for a logical Sale into web\_sales table |
| **ws\_sk** | **1** |
| ws\_sold\_date\_sk | 1 |
| ws\_sold\_time\_sk, | 1 |
| ws\_ship\_date\_sk, | 1 |
| ws\_item\_sk, | N |
| ws\_bill\_customer\_sk, | 1 |
| ws\_bill\_cdemo\_sk, | 1 |
| ws\_bill\_hdemo\_sk, | 1 |
| ws\_bill\_addr\_sk, | 1 |
| ws\_ship\_customer\_sk, | 1 |
| ws\_ship\_cdemo\_sk, | 1 |
| ws\_ship\_hdemo\_sk, | 1 |
| ws\_ship\_addr\_sk, | 1 |
| ws\_web\_page\_sk, | 1 |
| ws\_web\_site\_sk, | 1 |
| ws\_ship\_mode\_sk, | 1 |
| ws\_warehouse\_sk, | 1 |
| ws\_promo\_sk, | 1 |
| ws\_order\_number, | 1 |
| ws\_quantity, | N |
| ws\_wholesale\_cost | N |
| ws\_list\_price | N |
| ws\_sales\_price | N |
| ws\_ext\_discount\_amt | N |
| ws\_ext\_sales\_price | N |
| ws\_ext\_wholesale\_cost | N |
| ws\_ext\_list\_price | N |
| ws\_ext\_tax | N |
| ws\_coupon\_amt | N |
| ws\_ext\_ship\_cost | N |
| ws\_net\_paid | N |
| ws\_net\_paid\_inc\_tax | N |
| ws\_net\_paid\_inc\_ship | N |
| ws\_net\_paid\_inc\_ship\_tax | N |
| ws\_net\_profit | N |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **web\_returns** | Type | NULL? | Table is used by **Queries**: | Description |
| wr\_returned\_date\_sk | BIGINT default 9999999 |  | Q19 | Random reference to date after! referenced web\_sales ws\_soled\_date\_sk with same order number |
| wr\_returned\_time\_sk | BIGINT |  |  | Random reference to time\_dim t\_time\_sk |
| wr\_item\_sk | BIGINT | NOT NULL | Q16 Q19 | Random [1-N] item\_sk’s from ws\_item\_sk’s in referenced web\_sales ws\_order\_number (not necessary only one or all items from a store\_sales ticket are returned) |
| wr\_refunded\_customer\_sk | BIGINT |  |  | Probability choice  ${WS\_GIFT\_PCT} : Random reference to table: customer  c\_customer\_sk  1 - ${WS\_GIFT\_PCT} : same as ws\_ship\_customer\_sk |
| wr\_refunded\_cdemo\_sk | BIGINT |  |  | same cdemo\_sk as referenced customer selected in wr\_refundedl\_customer\_sk |
| wr\_refunded\_hdemo\_sk | BIGINT |  |  | same hdemo\_sk as referenced customer selected in wr\_refundedl\_customer\_sk |
| wr\_refunded\_addr\_sk | BIGINT |  |  | same addr\_sk as referenced customer selected in wr\_refundedl\_customer\_sk |
| wr\_returning\_customer\_sk | BIGINT |  |  | Same as wr\_refundedl\_customer\_sk |
| wr\_returning\_cdemo\_sk | BIGINT |  |  | Same as wr\_refunded\_cdemo\_sk |
| wr\_returning\_hdemo\_sk | BIGINT |  |  | Same as wr\_refunded\_hdemo\_sk |
| wr\_returning\_addr\_sk | BIGINT |  |  | Same as wr\_refunded\_addr\_sk |
| wr\_web\_page\_sk | BIGINT |  |  | Reference to ws\_web\_page\_sk, same as in web\_sales with same order\_number |
| wr\_reason\_sk | BIGINT |  |  | random Reference to reason r\_reason\_sk for every returned item |
| wr\_order\_number | BIGINT |  | Q16 | Reference a uniqe existing order\_number from web\_sales ws\_order\_number |
| wr\_return\_quantity | INTEGER |  | Q19 | Random number of returned pieces for every returned sr\_item\_sk:  Random[1, ss\_quantity] |
| wr\_return\_amt | DECIMAL (7 ,2) |  |  | ws\_sales\_price \* wr\_return\_quantity |
| wr\_return\_tax | DECIMAL (7 ,2) |  |  | wr\_return\_amt \* tax\_pct  with tax\_pct = random decimal between [0.00, 0.09] |
| wr\_return\_amt\_inc\_tax | DECIMAL (7 ,2) |  |  | wr\_return\_amt + wr\_return\_tax |
| wr\_fee | DECIMAL (7 ,2) |  |  | Random decimal between [0.50, 100.00] |
| wr\_return\_ship\_cost | DECIMAL (7 ,2) |  |  | ws\_list\_price \* random[0.00, 1.00] \* wr\_return\_quantity |
| wr\_refunded\_cash | DECIMAL (7 ,2) |  | Q16 | rand[0.0,1.0] \* wr\_return\_amt |
| wr\_reversed\_charge | DECIMAL (7 ,2) |  |  | rand[0.01, 1.00] \* (wr\_return\_amt -wr\_refunded\_cash) |
| wr\_account\_credit | DECIMAL (7 ,2) |  |  | wr\_return\_amt -wr\_reversed\_charge - wr\_refunded\_cash |
| wr\_net\_loss | DECIMAL (7 ,2) |  |  | wr\_return\_amt + wr\_return\_ship\_cost + wr\_return\_tax - wr\_store\_credit - wr\_refunded\_cash - wr\_reversed\_charge + wr\_fee |
| DISTRIBUTE BY HASH ( wr\_item\_sk ); |  |  |  |  |

**web\_returns**

${web\_returns\_size} = ${return\_web\_sale\_likelihood} \* ${web\_sales\_size}

web\_returns contains returned items for web\_sales. A logical web\_sale is identified by ws\_ticket\_id. This table must not contain more the one logical return entry for the same ws\_order\_number.

Return not all but random 1-N items from a selected web\_sale. Like in web\_sales, one logical “web\_return” contains multiple items and prodcues a web\_sales table row per returned item.

|  |  |
| --- | --- |
| **Logical sale** | Pick a random unique web\_sale ws\_order\_number. The selected web\_sale consists of N items. From these N items return random M items.  M=random[1, N]  1=same for every M  Write M lines for a logical return into web\_returns table |
| wr\_returned\_date\_sk, | 1 |
| wr\_returned\_time\_sk, | 1 |
| wr\_item\_sk, | M |
| wr\_refunded\_customer\_sk, | 1 |
| wr\_refunded\_cdemo\_sk | 1 |
| wr\_refunded\_hdemo\_sk | 1 |
| wr\_refunded\_addr\_sk | 1 |
| wr\_returning\_customer\_sk | 1 |
| wr\_returning\_cdemo\_sk | 1 |
| wr\_returning\_hdemo\_sk | 1 |
| wr\_returning\_addr\_sk | 1 |
| wr\_web\_page\_sk | M |
| wr\_reason\_sk | M |
| wr\_order\_number | 1 |
| wr\_return\_quantity | M |
| wr\_return\_amt, | M |
| wr\_return\_tax | M |
| wr\_return\_amt\_inc\_tax | M |
| wr\_fee | M |
| wr\_return\_ship\_cost | M |
| wr\_refunded\_cash | M |
| wr\_reversed\_charge | M |
| wr\_account\_credit | M |
| wr\_net\_loss | M |

${clickstreams\_chunksize} \* ${web\_sales\_size}

Web-clickstream contains information about each click (e.g. clicking on a link on a webpage) during a visitorˈs session.

Every visitor generates a “chunk” of n-lines with the same wcs\_click\_sk in the web\_clickstreams table. The table lines of each “chunk” are not continuous but interleaved with lines from other “chunks” (as they would be in a real “clickstream” log file as seen by the webserver).

Every clickstream “chunk” consists of multiple clicks with a total between: random [mean\_clicks\_per\_visitor-1, mean\_clicks\_per\_visitor+5].

Depending on the user type (buyer/visitor), there are different associated paths and data fields.

20% of all clicks are “buyers”. Buyers are registered users with a user\_sk and a buy has an associated sales\_sk. User\_sk and sales\_sk are linked to corresponding entries from the web\_sales table. Obviously, every item bought in web\_sales was “clicked” by a user. In additon to the items bought, a user may have clicked on addional rand[${mean\_clicks\_per\_buyer}-1, ${mean\_clicks\_per\_buyer}+2] items he or she only viewed. It is important that the implicit referential integrity between web\_sales and web\_clickstreams is consistent.

80% of all clicks are “visitors”. A visitor clickstream does not end in a purchase. Nevertheless, a “visitor” can still be a logged in user with an associated user\_sk. 50% of the visitors are logged in users and 50% are anonymous. Both, known and anonymous users, share the same behavior of doing rand[${mean\_clicks\_per\_buyer}-5, ${mean\_clicks\_per\_buyer}+5] clicks during their session.

Associated data fields

0,2

0,8

0,5

0,5

Items into shopping chart: (same items as in associated web\_sale)

Min: ${WS\_ITEMS\_PER\_ORDER\_MIN}

Max: ${WS\_ITEMS\_PER\_ORDER\_MAX}

not bought items, Random:

Min:${mean\_clicks\_per\_buyer} – 1,

Man:${mean\_clicks\_per\_buyer} +2

not bought items, Random:

Min:${mean\_clicks\_per\_buyer} – 5,

Man:${mean\_clicks\_per\_buyer} +5

Sales\_sk

User\_sk

Item\_sk

Web\_page\_sk

<<Per click>>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **web\_clickstreams** | Type | NULL? | Table is used by **Queries**: | Description |
| **wcs\_click\_sk** | BIGINT | NOT NULL |  |  |
| wcs\_click\_date\_sk | BIGINT |  | Q3 Q4 Q8 Q12 | Probability choice:  ${visitor\_likelihood} : Visitor  1- ${visitor\_likelihood} : Buyer  Case Visitor:  ${web\_sales\_days\_since\_date\_begin\_date} + Math.floor( (current ID or row) \* ((${web\_sales\_days\_within} - 1) / ${web\_clickstreams\_size}))  Case Buyer:  The clickstream must have the same reference to web\_sales\_ws\_sold\_date\_sk as the associate web\_sale (choosen by: wcs\_user\_sk) |
| wcs\_click\_time\_sk | BIGINT |  | Q3 Q4 Q8 Q12 | Probability choice: same choice as wcs\_click\_date\_sk,  ${visitor\_likelihood} : Visitor  1- ${visitor\_likelihood} : Buyer  Case Visitor:  Random referece to time\_dim t\_time\_sk  Case Buyer:  Random referece to web\_sales\_ws\_sold\_time\_sk |
| wcs\_sales\_sk | BIGINT |  | Q3 Q8 | Probability choice: same choice as wcs\_click\_date\_sk,  ${visitor\_likelihood} : Visitor  1- ${visitor\_likelihood} : Buyer  Case Visitor:  Value: “”  Case Buyer:  (Current row or id) \* 1 / ${clickstreams\_chunksize} |
| wcs\_item\_sk | BIGINT | can be null | Q2 Q3 Q4 Q5 Q8 Q12 Q30 | Probability choice: same choice as wcs\_click\_date\_sk,  ${visitor\_likelihood} : Visitor  1- ${visitor\_likelihood} : Buyer  Case Visitor:  Random referece to item i\_item\_sk  Case Buyer:  The clickstream must contain all ws\_item\_sk from the associated web\_sale (choosen by wcs\_user\_sk) plus additional random[${mean\_clicks\_per\_buyer} – 1, ${mean\_clicks\_per\_buyer} +2] clicked items (random references to item i\_item\_sk) which where not purchased. |
| wcs\_web\_page\_sk | BIGINT |  | Q8 | Probability choice: same choice as wcs\_click\_date\_sk,  ${visitor\_likelihood} : Visitor  1- ${visitor\_likelihood} : Buyer  Case Visitor:  Random referece to web\_page wp\_web\_page\_sk  Case Buyer:  The clickstream must contain all ws\_web\_page\_sk’s from the associated web\_sale plus additional random[${mean\_clicks\_per\_buyer} – 1, ${mean\_clicks\_per\_buyer} +2] clicked ws\_web\_page\_sk’s (random references to web\_page wp\_web\_page\_sk) .  One random web\_page\_sk for every random wcs\_item sk (same random choice as wcs\_item\_sk) |
| wcs\_user\_sk | BIGINT | can be null | Q2 Q3 Q4 Q5 Q8 Q12 Q30 | Probability choice: same choice as wcs\_click\_date\_sk,  ${visitor\_likelihood} : Visitor  1- ${visitor\_likelihood} : Buyer  Case Visitor:  Probability choice:  ${visitor\_known\_likelihood}: known visitor  1- ${visitor\_known\_likelihood} :unknown visitor  Case Buyer:  Choose a buying user from ws\_user\_sk  Note: wcs\_click\_date\_sk, wcs\_item\_sk and wcs\_web\_page\_sk must reflect the values from the associated web\_sale (purchasing multiple items in one “clickstream-session”): ws\_user\_sk, ws\_click\_date\_sk, ws\_item\_sk and ws\_web\_page\_sk |
| Notes |  |  |  |  |
| DISTRIBUTE BY HASH ( wcs\_click\_sk ); |  |  |  |  |

**warehouse**

${warehouse\_size} =5.0d \* ${SF\_log\_5}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **warehouse** | Type | NULL? | Table is used by **Queries**: | Description |
| w\_warehouse\_sk | BIGINT | NOT NULL | Q16 Q23 | Dense unique sequence. Starts at ${SK\_ID\_OFFSET} |
| w\_warehouse\_id | CHAR (16) | NOT NULL |  | Unique String, len: 16  charset: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”  Example: AAAAAAAAOKJNECAA |
| w\_warehouse\_name | VARCHAR (20) |  | Q22 Q23 | Text (multiple words)  len(min/max): 5 |
| w\_warehouse\_sq\_ft | INTEGER |  |  | Unifrom between ${W\_SQFT\_MIN}, ${W\_SQFT\_MAX} |
| w\_street\_number | CHAR (10) |  |  | DIST\_UNIFORM, 1, 1000, |
| w\_street\_name | VARCHAR (60) |  |  | Probability:  50% 1 word “%s”  50% 2 Words “%s %s”  From Weighted list "street\_names", valueCol:0 weightCoL 0 |
| w\_street\_type | CHAR (15) |  |  | Weighted list "street\_type", valueCol:0 weightCoL 0, |
| w\_suite\_number | CHAR (10) |  |  | Fromat: “Suite %d" DIST\_UNIFORM, 1, 100  suite number is alphabetic 50% of the time |
| w\_city | VARCHAR (60) |  |  | ity is picked from a distribution which maps to large/medium/small  Weighted list "cities". Value:0 weight:5 |
| w\_county | VARCHAR (30) |  |  | Weighted list "fips\_county" value column “county”,  choose a “region” use same region for all cols: county, state, zip, country, gmt\_offset |
| w\_state | CHAR (2) |  | Q16 | Weighted list "fips\_county” value column “st”  match region and country |
| w\_zip | CHAR (10) |  |  | Random number [10000, 99999] |
| w\_country | VARCHAR (20) |  |  | Allways "United States" |
| w\_gmt\_offset | DECIMAL (5 ,2) |  |  | Weighted list "fips\_county” value column “gmt”  match state and county |
| Notes: |  |  |  |  |
| DISTRIBUTE BY REPLICATION ; |  |  |  |  |

**web\_page**

${web\_page\_size} =60 \* ${SF\_log\_1.5}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **web\_page** | Type | NULL? | Table is used by **Queries**: | Description |
| **wp\_web\_page\_sk** | BIGINT | NOT NULL | Q4 Q8 Q14 | Dense unique sequence. Starts at ${SK\_ID\_OFFSET} |
| wp\_web\_page\_id | CHAR (16) | NOT NULL |  | Unique String, len: 16  charset: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”  Example: AAAAAAAAOKJNECAA |
| wp\_rec\_start\_date | DATE |  |  | Date  from: ${web\_page\_begin\_date}  to: ${web\_sales\_begin\_date}  Format: yyyy-MM-dd  With: wp\_rec\_start\_date{n} < wp\_rec\_start\_date{n+1}  where n= **wp\_web\_page\_sk** |
| wp\_rec\_end\_date | DATE |  |  | No end date for the moment. Value: “”  Elese:  50% Empty 50%: wp\_rec\_start\_date + rand[2years, 4years] |
| wp\_creation\_date\_sk | BIGINT |  |  | Random reference to table: date d\_date\_sk |
| wp\_access\_date\_sk | BIGINT |  |  | Random reference to table: date d\_date\_sk  Else: wp\_rec\_access\_date >= wp\_rec\_creation\_date |
| wp\_autogen\_flag | CHAR (1) |  |  | Propability : value  ${WP\_AUTOGEN\_PCT}: 1  1-${WP\_AUTOGEN\_PCT}: 0 |
| wp\_customer\_sk | BIGINT |  |  | Random reference to table: customer c\_customer\_sk |
| wp\_url | VARCHAR (100) |  |  | “http://www.” + RANDOMSTRING\_[4, 85] +“.com” |
| wp\_type | CHAR (50) |  | Q4 Q8 | Weighted list " web\_page\_use” value column “0” |
| wp\_char\_count | INTEGER |  | Q14 | Radom integer between:  min =wp\_link\_count \* 125 + wp\_image\_count \* 50  max =wp\_link\_count \* 300 + wp\_image\_count \* 150 |
| wp\_link\_count | INTEGER |  |  | Random integer between: [${WP\_LINK\_MIN}, ${WP\_LINK\_MIN}] |
| wp\_image\_count | INTEGER |  |  | Random integer between: [${WP\_IMAGE\_MIN}, ${WP\_IMAGE\_MIN}] |
| wp\_max\_ad\_count | INTEGER |  |  | Random integer between: [${WP\_AD\_MIN}, ${WP\_AD\_MIN}] |
| Note: |  |  |  |  |
| DISTRIBUTE BY REPLICATION ; |  |  |  |  |

**web\_site**

(UNUSED/UNREFERENCED) only ref: web\_sales

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **web\_site** | Type | NULL? | Table is used by querys: | Description |
| **web\_site\_sk** | BIGINT | NOT NULL |  | Dense unique sequence. Starts at ${SK\_ID\_OFFSET} referenced by web\_sales |
| web\_site\_id | CHAR (16) | NOT NULL |  | Unique String, len: 16  charset: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”  Example: AAAAAAAAOKJNECAA |
| web\_rec\_start\_date | DATE |  |  | Date  from: 1997-08-16  to: 2001-08-16  Format: yyyy-MM-dd  With: iweb\_rec\_start\_date {n} < web\_rec\_start\_date {n+1}  where n= **web\_site\_sk** |
| web\_rec\_end\_date | DATE |  |  | No end date for the moment. Value: “”  Elese:  50% Empty 50%: wp\_rec\_start\_date + rand[2years, 4years] |
| web\_name | VARCHAR (50) |  |  | Template: „site\_%d“ with %d = current\_row |
| web\_open\_date\_sk | BIGINT |  |  | Random reference to date table d\_date\_sk |
| web\_close\_date\_sk | BIGINT |  |  | Radom d\_date\_sk > web\_open\_date\_sk |
| web\_class | VARCHAR (50) |  |  | Value: "Unknown" |
| web\_manager | VARCHAR (40) |  |  | Pattern: "%s %s"  Weighted List ‘first\_names’, value\_col= 0; weight\_col: 0  Weighted List ‘last\_names’; , value\_col= 0; weight\_col: 0 |
| web\_mkt\_id | INTEGER |  |  | Random integer between: [1, 6] |
| web\_mkt\_class | VARCHAR (50) |  |  | Sentences following pseudo englisch gramatic  Example:  Clear circumstances know then further white companies. Typical budgets take both required children. Appeals must not make civil, financial representatives. Emotional areas shall wear only. |
| web\_mkt\_desc | VARCHAR (100) |  |  | Sentences following pseudo englisch gramatic  Example:  Clear circumstances know then further white companies. Typical budgets take both required children. Appeals must not make civil, financial representatives. Emotional areas shall wear only. |
| web\_market\_manager | VARCHAR (40) |  |  | Pattern: "%s %s"  Weighted List ‘first\_names’, value\_col= 0; weight\_col: 0  Weighted List ‘last\_names’; , value\_col= 0; weight\_col: 0 |
| web\_company\_id | INTEGER |  |  | Random integer between: [1, 6] |
| web\_company\_name | CHAR (50) |  |  | One random word from Weighted List ‘syllables’ |
| web\_street\_number | CHAR (10) |  |  | Address like in warehouse |
| web\_street\_name | VARCHAR (60) |  |  | Address like in warehouse |
| web\_street\_type | CHAR (15) |  |  | Address like in warehouse |
| web\_suite\_number | CHAR (10) |  |  | Address like in warehouse |
| web\_city | VARCHAR (60) |  |  | Address like in warehouse |
| web\_county | VARCHAR (30) |  |  | Address like in warehouse |
| web\_state | CHAR (2) |  |  | Address like in warehouse |
| web\_zip | CHAR (10) |  |  | Address like in warehouse |
| web\_country | VARCHAR (20) |  |  | Address like in warehouse |
| web\_gmt\_offset | DECIMAL (5 ,2) |  |  | Address like in warehouse |
| web\_tax\_percentage | DECIMAL (5 ,2) |  |  | Random decimal betweem [0.00, 0.12] |
| Notes: |  |  |  |  |
| DISTRIBUTE BY REPLICATION ; |  |  |  |  |

**reason**

only referenced by sr\_reason\_sk and wr\_reason\_sk (both not used in **Queries**)

size: 35 \* ${SF\_log\_1.5}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **reason** | Type | NULL? | Table is used by **Queries**: | Description  Example:  1|AAAAAAAABAAAAAAA|Package was damaged| |
| **r\_reason\_sk** | BIGINT | NOT NULL |  | Dense unique sequence. Starts at ${SK\_ID\_OFFSET} |
| r\_reason\_id | CHAR (16) | NOT NULL |  | Unique String, len: 16  charset: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”  Example: AAAAAAAAOKJNECAA |
| r\_reason\_desc | CHAR (100) |  |  | Weighted List ‘return\_reasons’, row = r\_reason\_sk; value\_col= 0; weight column: 0 |
| Notes. |  |  |  |  |
| DISTRIBUTE BY REPLICATION ; |  |  |  |  |

**ship\_mode**

(UNUSED/UNREFERENCED)

size: fixed size of 20

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ship\_mode** | Type | NULL? | Table is used by querys: | Description  Example:  1|AAAAAAAABAAAAAAA|EXPRESS|AIR|UPS|YvxVaJI10| |
| **sm\_ship\_mode\_sk** | BIGINT | NOT NULL |  | Dense unique sequence. Starts at ${SK\_ID\_OFFSET} |
| sm\_ship\_mode\_id | CHAR (16) | NOT NULL |  | Unique String, len: 16  charset: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”  Example: AAAAAAAAOKJNECAA |
| sm\_type | CHAR (30) |  |  | Weighted list “ship\_mode". Value:0 weight:0 |
| sm\_code | CHAR (10) |  |  | Weighted list “ship\_mode\_code". Value:0 weight:0 |
| sm\_carrier | CHAR (20) |  |  | Weighted list “ship\_mode\_carrier ". Value:0 weight:0 |
| sm\_contract | CHAR (20) |  |  | RandString ALPHANUM,  min/max: RS\_SM\_CONTRACT, SM\_CONTRACT |
| Notes: |  |  |  |  |
| DISTRIBUTE BY REPLICATION ; |  |  |  |  |

**income\_band**

(NOT USED!)

size: fixed 20

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **income\_band** | Type | NULL? | Table is used by querys: | Description  Example:  1|0|10000| |
| **ib\_income\_band\_sk,** | BIGINT | NOT NULL |  | Dense unique sequence. Starts at ${SK\_ID\_OFFSET} |
| ib\_lower\_bound | INTEGER |  |  | Weighted List ‘income\_band’ ; row= ib\_income\_band\_sk, valueCol=0 |
| ib\_upper\_bound | INTEGER |  |  | Weighted List ‘income\_band’ ; row= ib\_income\_band\_sk, valueCol=1 |
| Note: |  |  |  |  |
| DISTRIBUTE BY REPLICATION ; |  |  |  |  |

* + 1. Data Generation

The data generator used is based on an extension of the Parallel Data Generation Framework (PDGF). PDGF is a parallel data generator that is capable of producing large amounts of data for an arbitrary schema. The existing PDGF can be used to generate the structured part of the BigBench model. However, it is not capable of generating the unstructured product reviews text. First, PDGF is enhanced to produce a key-value data set for a fixed set of required and optional keys. This is sufficient to generate the weblogs part of BigBench.

The main challenge in product reviews is producing the unstructured text. This is achieved by an algorithm that produces synthetic text based on sample input text. The algorithm uses a Markov Chain technique that extracts key words and builds a dictionary based on these key words. The new algorithm is applied for BigBench by using some real product reviews from an online retailer for the initial sample data. PDGF interacts with the review generator through an API sending a product category as input and receiving a product review text for that category.

The volume dimension model is far simpler than the variety discussion and previous data generators had a good handle on that. PDGF handles the volume well since it can scale the size of the data based on a scale factor. It also runs efficiently for large scale factors since it runs in parallel and can leverage large systems dedicated for the benchmark.

* 1. Query Overview

This section illustrates a high level overview of the 30 **Queries** of BigBench. It is structured into a general overview of the different **Query** types, a textual description of the 30 **Queries** as well as specific characteristics of implementation.

* + 1. Query types

The **Queries** used in BigBench can be grouped into three categories: Structured, semi-structured and unstructured. The following table illustrates the data types that the **Queries** access as specified in Clause B.1.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Structured | 16 | Structured |
| 2 | Semi-Structured | 17 | Structured |
| 3 | Semi-Structured | 18 | Un-Structured |
| 4 | Semi-Structured | 19 | Un-Structured |
| 5 | Semi-Structured | 20 | Structured |
| 6 | Structured | 21 | Structured |
| 7 | Structured | 22 | Structured |
| 8 | Semi-Structured | 23 | Structured |
| 9 | Structured | 24 | Structured |
| 10 | Un-Structured | 25 | Structured |
| 11 | Structured | 26 | Structured |
| 12 | Semi-Structured | 27 | Un-Structured |
| 13 | Structured | 28 | Un-Structured |
| 14 | Structured | 29 | Structured |
| 15 | Structured | 30 | Semi-Structured |

* + 1. Query Grouping

The overall number of the thirty **Queries** has been grouped into four categories: Pure Hive **Queries**, Hive **Queries** with MapReduce programs, Hive **Queries** using natural language processing, and **Queries** using Apache Spark MLLIB. In the following, an example for each of the different flavors of **Queries** will be given. The distribution of the different **Query** types is shown in the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case** | **Method** | **Use case** | **Method** |
| 1 | UDF/UDTF | 16 | Pure QL |
| 2 | Map Reduce | 17 | Pure QL |
| 3 | Map Reduce | 18 | UDF/UDTF/NLP |
| 4 | Map Reduce | 19 | UDF/UDTF/NLP |
| 5 | ML | 20 | ML |
| 6 | Pure QL | 21 | Pure QL |
| 7 | Pure QL | 22 | Pure QL |
| 8 | Map Reduce | 23 | Pure QL |
| 9 | Pure QL | 24 | Pure QL |
| 10 | UDF/UDTF/NLP | 25 | ML |
| 11 | Pure QL | 26 | ML |
| 12 | Pure QL | 27 | UDF/UDTF/NLP |
| 13 | Pure QL | 28 | ML |
| 14 | Pure QL | 29 | UDF/UDTF |
| 15 | Pure QL | 30 | UDF/UDTF/Map Reduce |

It should be noted that **Queries** that use NLTK and Mahout also require preprocessing by Hive. Therefore, Apache Hive is critical to all data processing activities in this implementation of BigBench.

* 1. Query Descriptions

This section gives a textual description of each **Query**.

**Query 01**

Find top 100 products that are sold together frequently in given stores. Only products in certain categories sold in specific stores are considered, and "sold together frequently" means at least 50 customers bought these productstogether in a transaction.

**Query 02**

Find the top 30 products that are mostly viewed together with a given product in online store. Note that the order of products viewed does not matter, and "viewed together" relates to a web\_clickstreams click\_session of a known user with a session timeout of 60min.If the duration between two click of a user is greater then the session timeout, a new session begins. With a session timeout of 60min.

**Query 03**

For a given product get a top 30 list sorted by number of views in descending order of the last 5 products that are mostly viewed before the product was purchased online. For the viewed products, consider only products in certain item categories and viewed within 10days before the purchase date.

**Query 04**

Web\_clickstream shopping cart abandonment analysis: For users who added products in their shopping carts but did not check out in the online store during their session, find the average number of pages they visited during their sessions. A "session" relates to a click\_session of a known user with a session time-out of 60min.If the duration between two clicks of a user is greater then the session time-out, a new session begins.

**Query 05**

Build a model using logistic regression for a visitor to an online store: based on existing users’ online activities (interest in items of different categories) and demographics. This model will be used to predict if the visitor is interested in a given item category. Output the precision, accuracy and confusion matrix of model.

Note: Randomly choose 90% of users for model creation. Remaining 10% will be used later as unknown visitors for prediction.

**Query 06**

Identifies customers shifting their purchase habit from store to web sales. Find customers who spend in relation more money in the second year following a given year in the web\_sales channel then in the store sales channel. Report customers details: first name, last name, their country of origin, login name and email address) and identify if they are preferred customer, for the top 100 customers with the highest increase in their second year web purchase ratio.

**Query 07**

List top 10 states in descending order with at least 10 customers who during a given month bought products with the price tag at least 20% higher than the average price of products in the same category.

**Query 08**

For online sales, compare the total sales monetary amount in which customers checked online reviews before making the purchase and that of sales in which customers did not read reviews. Consider only online sales for a specific category in a given year.

**Query 09**

Aggregate total amount of sold items over different given types of combinations of customers based on selected groups of marital status, education status, sales price and different combinations of state and sales profit.

**Query 10**

For all products, extract sentences from its product reviews that contain positive or negative sentiment and display for each item the sentiment polarity of the extracted sentences (POS OR NEG) and the sentence and word in sentence leading to this classification.

**Query 11**

For a given product, measure the correlation of sentiments, including the number of reviews and average review ratings, on product monthly revenues within a given time frame.

**Query 12**

Find all customers who viewed items of a given category on the web in a given month and year that was followed by an instore purchase of an item from the same category in the three consecutive months.

**Query 13**

Display customers with both store and web sales in consecutive years for whom the increase in web sales exceeds the increase in store sales for a specified year.

**Query 14**

What is the ratio between the number of items sold over the internet in the morning (7 to 8am) to the number of items sold in the evening (7 to 8pm) of customers with a specified number of dependents. Consider onlywebsites with a high amount of content.

**Query 15**

Find the categories with flat or declining sales for in store purchases during a given year for a given store.

**Query 16**

Compute the impact of an item price change on the store sales by computing the total sales for items in a 30 day period before and after the price change. Group the items by location of warehouse where they were delivered from.

**Query 17**

Find the ratio of items sold with and without promotions in a given month and year. Only items in certain categories sold to customers living in a specific time zone are considered.

**Query 18**

Identify the stores with flat or declining sales in 4 consecutive months, check if there are any negative reviews regarding these stores available online.

**Query 19**

Retrieve the items with the highest number of returns where the number of returns was approximately equivalent across all store and web channels (within a tolerance of +/ 10%), within the week ending given dates. Analyse the online reviews for these items to see if there are any negative reviews.

**Query 20**

Customer segmentation for return analysis: Customers are separated along the following dimensions: return frequency, return order ratio (total number of orders partially or fully returned versus the total number of orders), return item ratio (total number of items returned versus the number of items purchased), return amount ration (total monetary amount of items returned versus the amount purchased), return order ratio. Consider the store returns during a given year for the computation.

**Query 21**

Get all items that were sold in stores in a given month and year and which were returned in the next 6 months and repurchased by the returning customer afterwards through the web sales channel in the following three years. For those items, compute the total quantity sold through the store, the quantity returned and the quantity purchased through the web. Group this information by item and store.

**Query 22**

For all items whose price was changed on a given date, compute the percentage change in inventory between the 30day period BEFORE the price change and the 30day period AFTER the change. Group this information by warehouse.

**Query 23**

This Query contains multiple, related iterations: Iteration 1: Calculate the coefficient of variation and mean of every item and warehouse of the given and the consecutive month. Iteration 2: Find items that had a coefficient of variation of 1.3 or larger in the given and the consecutive month

**Query 24**

For a given product, measure the effect of competitor's prices on products' instore and online sales. Compute the crossprice elasticity of demand for a given product.

**Query 25**

Customer segmentation analysis: Customers are separated along the following key shopping dimensions: recency of last visit, frequency of visits and monetary amount. Use the store and online purchase data during a given year to compute. After model of separation is build, report for the analysed customers to which "group" they where assigned.

**Query 26**

Cluster customers into book buddies/club groups based on their in store book purchasing histories. After model of separation is build, report for the analysed customers to which "group" they where assigned.

**Query 27**

For a given product, find "competitor" company names in the product reviews. Display review id, product id, "competitor’s" company name and the related sentence from the online review

**Query 28**

Build text classifier for online review sentiment classification (Positive, Negative, Neutral), using 90% of available reviews for training and the remaining 10% for testing. Display classifier accuracy on testing data and classification result for the 10% testing data: <reviewSK>,<originalRating>,<classificationResult>.

**Query 29**

Perform category affinity analysis for products purchased together online. Note that the order of products viewed does not matter,

**Query 30**

Perform category affinity analysis for products viewed together online. Note that the order of products viewed does not matter, and "viewed together" relates to a click\_session of a user with a session timeout of 60min. If the duration between two clicks of a user is greater then the session timeout, a new session begins.

* + 1. Schema

In the following, the complete schema definition for TPCx-BB Hive is listed in Appendix I

* + 1. Weighted lists

See files: weightedList\_probabilities.txt and productReviews\_weighted\_list\_probabilities.txt.

1. -- Query Parameters

Query Parameters

-- !echo ============================;

-- !echo <settings from queryParameters.sql>;

-- !echo ============================;

--new (dates all Mondays, dateranges complete weeks):

--store: 2000-01-03, 2004-01-05 (1463 days, 209 weeks)

--item: 2000-01-03, 2004-01-05 (1463 days, 209 weeks)

--web\_page: 2000-01-03, 2004-01-05 (1463 days, 209 weeks)

--store\_sales: 2001-01-01, 2006-01-02 (1827 days, 261 weeks)

--web\_sales: 2001-01-01, 2006-01-02 (1827 days, 261 weeks)

--inventory: 2001-01-01, 2006-01-02 (1820 days, 261 weeks)

-- READ ME

-- ITEM\_SK

-- Datagenerator ensures that item\_sk's 10000-10002 are very frequent accross all scalefactors

-------- Q01 -----------

--category\_ids:

--1 Home & Kitchen

--2 Music

--3 Books

--4 Clothing & Accessories

--5 Electronics

--6 Tools & Home Improvement

--7 Toys & Games

--8 Movies & TV

--9 Sports & Outdoors

set q01\_i\_category\_id\_IN=1, 2 ,3;

-- sf1 -> 11 stores, 90k sales in 820k lines

set q01\_ss\_store\_sk\_IN=10, 20, 33, 40, 50;

set q01\_viewed\_together\_count=50;

set q01\_limit=100;

-------- Q02 -----------

-- q02\_pid1\_IN=<pid>, <pid>, ..

--pid == item\_sk

--sf 1 item count: 17999c

set q02\_item\_sk=10001;

set q02\_MAX\_ITEMS\_PER\_BASKET=5000000;

set q02\_limit=30;

set q02\_session\_timeout\_inSec=3600;

-------- Q03 -----------

set q03\_days\_in\_sec\_before\_purchase=864000;

set q03\_views\_before\_purchase=5;

set q03\_purchased\_item\_IN=10001;

--see q1 for categories

set q03\_purchased\_item\_category\_IN=2,3;

set q03\_limit=30;

-------- Q04 -----------

set q04\_session\_timeout\_inSec=3600;

-------- Q05 -----------

set q05\_i\_category='Movies & TV';

set q05\_cd\_education\_status\_IN='Advanced Degree', 'College', '4 yr Degree', '2 yr Degree';

set q05\_cd\_gender='M';

-------- Q06 -----------

SET q06\_LIMIT=100;

--web\_sales and store\_sales date

SET q06\_YEAR=2001;

-------- Q07 -----------

SET q07\_HIGHER\_PRICE\_RATIO=1.2;

--store\_sales date

SET q07\_YEAR=2004;

SET q07\_MONTH=7;

SET q07\_HAVING\_COUNT\_GE=10;

SET q07\_LIMIT=10;

-------- Q08 -----------

-- web\_clickstreams date range

set q08\_startDate=2001-09-02;

-- + 1year

set q08\_endDate=2002-09-02;

-- 3 days in sec = 3\*24\*60\*60

set q08\_seconds\_before\_purchase=259200;

-------- Q09 -----------

--store\_sales date

set q09\_year=2001;

set q09\_part1\_ca\_country=United States;

set q09\_part1\_ca\_state\_IN='KY', 'GA', 'NM';

set q09\_part1\_net\_profit\_min=0;

set q09\_part1\_net\_profit\_max=2000;

set q09\_part1\_education\_status=4 yr Degree;

set q09\_part1\_marital\_status=M;

set q09\_part1\_sales\_price\_min=100;

set q09\_part1\_sales\_price\_max=150;

set q09\_part2\_ca\_country=United States;

set q09\_part2\_ca\_state\_IN='MT', 'OR', 'IN';

set q09\_part2\_net\_profit\_min=150;

set q09\_part2\_net\_profit\_max=3000;

set q09\_part2\_education\_status=4 yr Degree;

set q09\_part2\_marital\_status=M;

set q09\_part2\_sales\_price\_min=50;

set q09\_part2\_sales\_price\_max=200;

set q09\_part3\_ca\_country=United States;

set q09\_part3\_ca\_state\_IN='WI', 'MO', 'WV';

set q09\_part3\_net\_profit\_min=50;

set q09\_part3\_net\_profit\_max=25000;

set q09\_part3\_education\_status=4 yr Degree;

set q09\_part3\_marital\_status=M;

set q09\_part3\_sales\_price\_min=150;

set q09\_part3\_sales\_price\_max=200;

-------- Q10 -----------

--no params

-------- Q11 -----------

--web\_sales date range

set q11\_startDate=2003-01-02;

-- +30days

set q11\_endDate=2003-02-02;

-------- Q12 -----------

--web\_clickstreams start\_date - endDate1

--store\_sales start\_date - endDate2

set q12\_startDate=2001-09-02;

set q12\_endDate1=2001-10-02;

set q12\_endDate2=2001-12-02;

set q12\_i\_category\_IN='Books', 'Electronics';

-------- Q13 -----------

--store\_sales date

set q13\_Year=2001;

set q13\_limit=100;

-------- Q14 -----------

set q14\_dependents=5;

set q14\_morning\_startHour=7;

set q14\_morning\_endHour=8;

set q14\_evening\_startHour=19;

set q14\_evening\_endHour=20;

set q14\_content\_len\_min=5000;

set q14\_content\_len\_max=6000;

-------- Q15 -----------

--store\_sales date range

set q15\_startDate=2001-09-02;

--+1year

set q15\_endDate=2002-09-02;

set q15\_store\_sk=10;

-------- Q16 -----------

-- web\_sales/returns date

set q16\_date=2001-03-16;

-------- Q17 -----------

set q17\_gmt\_offset=-5;

--store\_sales date

set q17\_year=2001;

set q17\_month=12;

set q17\_i\_category\_IN='Books', 'Music';

-------- Q18 -----------

-- store\_sales date range

set q18\_startDate=2001-05-02;

--+90days

set q18\_endDate=2001-09-02;

-------- Q19 -----------

set q19\_storeReturns\_date\_IN='2004-03-08' ,'2004-08-02' ,'2004-11-15', '2004-12-20';

set q19\_webReturns\_date\_IN='2004-03-08' ,'2004-08-02' ,'2004-11-15', '2004-12-20';

set q19\_store\_return\_limit=100;

-------- Q20 -----------

--no params

-------- Q21 -----------

--store\_sales/returns web\_sales/returns date

-- ss\_date\_sk range at SF 1

--36890 2001-01-01

--38697 2005-12-13

set q21\_year=2003;

set q21\_month=1;

set q21\_limit=100;

-------- Q22 -----------

--inventory date

set q22\_date=2001-05-08;

set q22\_i\_current\_price\_min=0.98;

set q22\_i\_current\_price\_max=1.5;

-------- Q23 -----------

--inventory date

set q23\_year=2001;

set q23\_month=1;

set q23\_coefficient=1.3;

-------- Q24 -----------

set q24\_i\_item\_sk=10000;

-------- Q25 -----------

-- store\_sales and web\_sales date

set q25\_date=2002-01-02;

-------- Q26 -----------

set q26\_i\_category\_IN='Books';

set q26\_count\_ss\_item\_sk=5;

-------- Q27 -----------

set q27\_pr\_item\_sk=10002;

-------- Q28 -----------

--no params

-------- Q29 -----------

set q29\_limit=100;

set q29\_session\_timeout\_inSec=3600;

-------- Q30 -----------

set q30\_limit=100;

set q30\_session\_timeout\_inSec=3600;

-- !echo ============================;

-- !echo </settings from queryParameters.sql>;

-- !echo ============================;

1. – Benchmark Parameters

– Benchmark generic Parameters.

## ==========================

## JAVA environment

## ==========================

export BIG\_BENCH\_JAVA="java"

## ==========================

## common query resources

## ==========================

export BIG\_BENCH\_QUERY\_RESOURCES="${BIG\_BENCH\_HOME}/distributions/Resources"

## ==========================

## default settings for benchmark

## ==========================

export BIG\_BENCH\_DEFAULT\_DATABASE="bigbench"

export BIG\_BENCH\_DEFAULT\_DISTRO\_LOCATION="cdh/6.0"

export BIG\_BENCH\_DEFAULT\_ENGINE="hive"

export BIG\_BENCH\_DEFAULT\_MAP\_TASKS="80"

export BIG\_BENCH\_DEFAULT\_SCALE\_FACTOR="2"

export BIG\_BENCH\_DEFAULT\_NUMBER\_OF\_PARALLEL\_STREAMS="2"

export BIG\_BENCH\_DEFAULT\_BENCHMARK\_PHASE="run\_query"

export BIG\_BENCH\_HDFS\_NAMENODE\_URI=""

export DEFAULT\_ENGINE\_ENV\_INFO\_FILE="logEngineEnvInfo"

## ==========================

## HADOOP environment

## ==========================

##folder containing the cluster setup \*-site.xml files like core-site.xml

export BIG\_BENCH\_HADOOP\_CONF="/etc/hadoop/conf.cloudera.hdfs"

export BIG\_BENCH\_HADOOP\_LIBS\_NATIVE="/opt/cloudera/parcels/CDH/lib/hadoop/lib/native"

## ==========================

## HDFS config and paths

## ==========================

export BIG\_BENCH\_USER="$USER"

export BIG\_BENCH\_HDFS\_ABSOLUTE\_PATH="${BIG\_BENCH\_HDFS\_NAMENODE\_URI}/user/$BIG\_BENCH\_USER" ##working dir of benchmark.

export BIG\_BENCH\_HDFS\_RELATIVE\_HOME="benchmarks/bigbench"

export BIG\_BENCH\_HDFS\_QUERY\_RESOURCES="${BIG\_BENCH\_HDFS\_RELATIVE\_HOME}/Resources"

export BIG\_BENCH\_HDFS\_RELATIVE\_INIT\_DATA\_DIR="$BIG\_BENCH\_HDFS\_RELATIVE\_HOME/data"

export BIG\_BENCH\_HDFS\_RELATIVE\_REFRESH\_DATA\_DIR="$BIG\_BENCH\_HDFS\_RELATIVE\_HOME/data\_refresh"

export BIG\_BENCH\_HDFS\_RELATIVE\_QUERY\_RESULT\_DIR="$BIG\_BENCH\_HDFS\_RELATIVE\_HOME/queryResults"

export BIG\_BENCH\_HDFS\_RELATIVE\_TEMP\_DIR="$BIG\_BENCH\_HDFS\_RELATIVE\_HOME/temp"

export BIG\_BENCH\_HDFS\_ABSOLUTE\_HOME="$BIG\_BENCH\_HDFS\_ABSOLUTE\_PATH/$BIG\_BENCH\_HDFS\_RELATIVE\_HOME"

export BIG\_BENCH\_HDFS\_ABSOLUTE\_INIT\_DATA\_DIR="$BIG\_BENCH\_HDFS\_ABSOLUTE\_PATH/$BIG\_BENCH\_HDFS\_RELATIVE\_INIT\_DATA\_DIR"

export BIG\_BENCH\_HDFS\_ABSOLUTE\_REFRESH\_DATA\_DIR="$BIG\_BENCH\_HDFS\_ABSOLUTE\_PATH/$BIG\_BENCH\_HDFS\_RELATIVE\_REFRESH\_DATA\_DIR"

export BIG\_BENCH\_HDFS\_ABSOLUTE\_QUERY\_RESULT\_DIR="$BIG\_BENCH\_HDFS\_ABSOLUTE\_PATH/$BIG\_BENCH\_HDFS\_RELATIVE\_QUERY\_RESULT\_DIR"

export BIG\_BENCH\_HDFS\_ABSOLUTE\_TEMP\_DIR="$BIG\_BENCH\_HDFS\_ABSOLUTE\_PATH/$BIG\_BENCH\_HDFS\_RELATIVE\_TEMP\_DIR"

## ==========================

## Data redundancy report

## ==========================

export BIG\_BENCH\_FILESYSTEM\_CHECK\_CMD="hdfs fsck -blocks"

export BIG\_BENCH\_DISK\_USAGE\_CHECK\_CMD="hdfs dfs -du -s -h ${BIG\_BENCH\_HDFS\_RELATIVE\_INIT\_DATA\_DIR}"

export BIG\_BENCH\_FILESYSTEM\_ECPOLICY\_CMD="hdfs ec -getPolicy -path ${BIG\_BENCH\_HDFS\_ABSOLUTE\_PATH}"

# --------------------------------------------

# Hadoop data generation options

# --------------------------------------------

# specify JVM arguments like: -Xmx2000m;

# default of: 800m is sufficient if the datagen only uses "-workers 1" - one worker thread per map task

# Add +100MB per additional worker if you modified: BIG\_BENCH\_DATAGEN\_HADOOP\_OPTIONS

export BIG\_BENCH\_DATAGEN\_HADOOP\_JVM\_ENV="$BIG\_BENCH\_JAVA -Xmx800m"

# if you increase -workers, you must also increase the -Xmx setting in BIG\_BENCH\_DATAGEN\_HADOOP\_JVM\_ENV;

#-ap:=automatic progress ,3000ms intervall; prevents hadoop from killing long running jobs. Datagen runs piggyback on a map task as external process. If the external process does not periodically send a keepalive on stdout, the map task can not signal to the task tracker it is still alive and making progress.

#-workers:=limit hadoop based data generator to use 1 CPU core per map task.

export BIG\_BENCH\_DATAGEN\_HADOOP\_OPTIONS=" -workers 1 -ap 3000 "

#replication count for staging data files written by the data generator during DATA\_GENERATION phase of the benchmark into HDFS directories:

#BIG\_BENCH\_HDFS\_ABSOLUTE\_INIT\_DATA\_DIR and BIG\_BENCH\_HDFS\_ABSOLUTE\_REFRESH\_DATA\_DIR

#recommended: =-1 -- use cluster default (typical HDFS default is =3)

# =1 -- to save space,

# =3 -- or any number you like

export BIG\_BENCH\_DATAGEN\_DFS\_REPLICATION="-1"

# if empty, generate all tables (default).

# Else: explicitly specify which tables to generate e.g.: BIG\_BENCH\_DATAGEN\_TABLES="item customer store"

# Tables to choose from: customer customer\_address customer\_demographics date\_dim household\_demographics income\_band inventory item item\_marketprices product\_reviews promotion reason ship\_mode store store\_returns store\_sales time\_dim warehouse web\_clickstreams web\_page web\_returns web\_sales web\_site

export BIG\_BENCH\_DATAGEN\_TABLES=""

# if distributed data generation fails, re run DATA\_GENERATION phase with BIG\_BENCH\_DATAGEN\_HADOOP\_EXEC\_DEBUG="-testDebugMessages" to retrieve more information on the cause. Dont forget to look into the yarn application and task logs!

export BIG\_BENCH\_DATAGEN\_HADOOP\_EXEC\_DEBUG=""

# the default behaviour is to stop the whole benchmark when an error occurs

# set this to 0 to keep on running (e.g. continue with next phase or query) when an error occurs

export BIG\_BENCH\_STOP\_AFTER\_FAILURE="1"

## Speed up HDFS operations like copy, move, delete, list, chmod, mkdir

## requires "snakebite" to be installed https://github.com/spotify/snakebite

## yum install epel-release

## yum install -y python-pip

## pip install snakebite

#0==off 1==on

export BIG\_BENCH\_USE\_SNAKEBITE\_HDFSCLIENT="0"

# set binary name of pssh for environment information gathering

# used to retrieve statistics and information from worker nodes

export BIG\_BENCH\_PSSH\_BINARY="pssh"

1. – Global Framework Parameters

--########### READ ME ################

-- The default way to set hive options is doing it globally for your whole cluster (e.g. cloudera manager, ambari, hive-site.xml, ...)

-- However, if for some reasons you cant or wont change your cluster global config, you can enable hive specific tuning options in this file.

-- Below are listed some commonly used settings. The values you see in this file may not apply to your own cluster! we used some of them on our 3 node (16cores 60gb ram) test instances

--#####################################

--###########################

-- EXECUTION ENGINE

--###########################

-- values: mr, tez, spark

-- set hive.execution.engine=mr;

-- ###########################

-- parallel order by. required by **Queries**:

-- ###########################

set bigbench.hive.optimize.sampling.orderby=true;

set bigbench.hive.optimize.sampling.orderby.number=20000;

set bigbench.hive.optimize.sampling.orderby.percent=0.1;

-- ###########################

-- output and itermediate table settings

-- ###########################

-- if you cluster has good cpu's but limited network bandwith, this could speed up the exchange of intermediate results (this option should be turund on if you cluster has high 'net wait i/o%'

-- set hive.exec.compress.intermediate=true;

-- set mapred.map.output.compression.codec=org.apache.hadoop.io.compress.SnappyCodec;

-- default is to keep the created result tables human readable.

-- set hive.exec.compress.output=false;

-- set mapred.output.compression.codec=org.apache.hadoop.io.compress.DefaultCodec;

-- set hive.default.fileformat=ORC;

-- ###########################

-- mappers settings

-- ###########################

-- Number of mappers used by HIVE, based on table sizes. If you experience underutilization or to much mappers/reducers, you can play with these settings

-- The number of physical files a table consists of is irrelevant for hives metric for estimating number of mappers. (Hive uses HiveCombineInputFormat, joining the files)

-- the following two parameters are most effective in influencing hives estimation of mappers. To low settings may result in to many map tasks, while to high size settings result in to few map tasks and underutilization of the cluster.

-- both extremes are harmful to the performance. For small data set sizes of 1-100GB a good value for max.split.size may be 134217728 (128MB). As an estimation, take a medium sized table and divide its size by the number of map tasks you need to utilize your cluster.

-- set mapreduce.input.fileinputformat.split.minsize=1048576;

-- set mapreduce.input.fileinputformat.split.maxsize=67108864;

-- ###########################

-- reducer settings

-- ###########################

-- Number of reducers used by HIVE

-- hives metric for estimating reducers is mostly controlled by the following settings. Node: Some Query functions like count(\*) or Distinct will lead to hive always using only 1 reducer

-- 1GB default

-- set hive.exec.reducers.bytes.per.reducer=33554432;

-- ###########################

-- optimizations for joins.

-- ###########################

-- things like mapjoins are done in memory and require a lot of it

-- README!

-- Hive 0.12 bug, hive ignores 'hive.mapred.local.mem' resulting in out of memory errors in map joins!

-- (more exactly: bug in Hadoop 2.2 where hadoop-env.cmd sets the -xmx parameter multiple times, effectively overriding the user set hive.mapred.local.mem setting. see: https://issues.apache.org/jira/browse/HADOOP-10245

-- There are 3 workarounds:

-- 1) assign more memory to the local!! Hadoop JVM client (not! mapred.map.memory)-> map-join child vm will inherit the parents jvm settings

-- 2) reduce "hive.smalltable.filesize" to ~1MB (depends on your cluster settings for the local JVM)

-- 3) turn off "hive.auto.convert.join" to prevent hive from converting the join to a mapjoin.

-- MAP join settings:

-- set hive.auto.convert.join.noconditionaltask.size=100000;

-- set hive.auto.convert.join=true;

-- set hive.optimize.mapjoin.mapreduce=true;

-- set hive.mapred.local.mem=1024;

-- default:25MB, max size of tables considered for local in memory map join. Beware! ORC files have only little file size but huge in memory data size! a 25MB ORC easily consumes 512MB.. related: https://issues.apache.org/jira/browse/HIVE-2601

-- set hive.mapjoin.smalltable.filesize=10000;

-- set hive.mapjoin.localtask.max.memory.usage=0.90;

-- set hive.auto.convert.sortmerge.join=true;

-- set hive.auto.convert.sortmerge.join.noconditionaltask=true;

-- set hive.auto.convert.join.noconditionaltask.size=100000;

-- set hive.optimize.bucketmapjoin=true;

-- set hive.optimize.bucketmapjoin.sortedmerge=false;

-- set hive.optimize.skewjoin=true; --READ FIRST: https://issues.apache.org/jira/browse/HIVE-5888

-- set hive.optimize.skewjoin.compiletime=true;

-- set hive.groupby.skewindata=true;

-- ###########################

-- Other tuning options

-- ###########################

-- exec.parallel is still considered unstable, but has the potential to increase you utilization by running multiple independent stages of a Query in parallel

-- set hive.exec.parallel=true;

-- set hive.exec.parallel.thread.number=8;

-- you should really turn these options on for your whole cluster, not just for bigbench

-- predicate pushdown for ORC-files (eager filtering of columns)

-- set hive.optimize.ppd=true;

-- set hive.optimize.ppd.storage=true;

-- set hive.ppd.recognizetransivity=false;

-- set hive.optimize.index.filter=true;

-- set hive.stats.autogather=true;

-- set hive.auto.convert.sortmerge.join=true;

-- set hive.vectorized.execution.enabled=true;

-- set hive.vectorized.execution.reduce.enabled=true;

-- set hive.cbo.enable=true;

-- set hive.compute.Query.using.stats=true;

-- set hive.stats.fetch.column.stats=true;

-- set hive.stats.fetch.partition.stats=true;

-- set hive.script.operator.truncate.env=true;

-- ============================;

-- Print most important properties;

-- ============================;

--exec engine and optimizer

set hive.execution.engine;

set hive.cbo.enable;

set hive.stats.fetch.partition.stats;

set hive.script.operator.truncate.env;

set hive.compute.Query.using.stats;

set hive.vectorized.execution.enabled;

set hive.vectorized.execution.reduce.enabled;

set hive.stats.autogather;

--input output

set mapreduce.input.fileinputformat.split.minsize;

set mapreduce.input.fileinputformat.split.maxsize;

set hive.exec.reducers.bytes.per.reducer;

set hive.exec.reducers.max;

set hive.exec.parallel;

set hive.exec.parallel.thread.number;

set hive.exec.compress.intermediate;

set hive.exec.compress.output;

set mapred.map.output.compression.codec;

set mapred.output.compression.codec;

set hive.default.fileformat;

--join optimizations

set hive.auto.convert.sortmerge.join;

set hive.auto.convert.sortmerge.join.noconditionaltask;

set hive.optimize.bucketmapjoin;

set hive.optimize.bucketmapjoin.sortedmerge;

set hive.auto.convert.join.noconditionaltask.size;

set hive.auto.convert.join;

set hive.optimize.mapjoin.mapreduce;

set hive.mapred.local.mem;

set hive.mapjoin.smalltable.filesize;

set hive.mapjoin.localtask.max.memory.usage;

set hive.optimize.skewjoin;

set hive.optimize.skewjoin.compiletime;

-- filter optimizations (predicate pushdown to storage level)

set hive.optimize.ppd;

set hive.optimize.ppd.storage;

set hive.ppd.recognizetransivity;

set hive.optimize.index.filter;

--other

set hive.optimize.sampling.orderby=true;

set hive.optimize.sampling.orderby.number;

set hive.optimize.sampling.orderby.percent;

set bigbench.hive.optimize.sampling.orderby;

set bigbench.hive.optimize.sampling.orderby.number;

set bigbench.hive.optimize.sampling.orderby.percent;

set hive.groupby.skewindata;

set hive.exec.submit.local.task.via.child;

-- Database - DO NOT DELETE OR CHANGE

CREATE DATABASE IF NOT EXISTS ${env:BIG\_BENCH\_DATABASE};

use ${env:BIG\_BENCH\_DATABASE};

1. – Local Settings Parameters

-- !echo ============================;

-- !echo <settings from engineLocalSettings.sql/conf>;

-- !echo ============================;

-------- Q01 ----------- Example only.

set hive.mapjoin.localtask.max.memory.usage = 3556

set hive.auto.convert.join = false

1. – SUT Hardware and Software

##############################

# Hardware #

##############################

##### /proc/cpuinfo #####

processor : 0-31

vendor\_id : GenuineIntel

cpu family : 6

model : 63

model name : Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz

stepping : 2

microcode : 37

cpu MHz : 2394.725

cache size : 30720 KB

physical id : 0

siblings : 20

core id : 0

cpu cores : 10

apicid : 0

initial apicid : 0

fpu : yes

fpu\_exception : yes

cpuid level : 13

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx rdtscp lm constant\_tsc rep\_good xtopology nonstop\_tsc aperfmperf unfair\_spinlock pni pclmulqdq monitor est ssse3 fma cx16 pcid sse4\_1 sse4\_2 x2apic movbe popcnt tsc\_deadline\_timer aes xsave avx f16c rdrand hypervisor lahf\_lm abm ida xsaveopt fsgsbase bmi1 avx2 smep bmi2 erms invpcid

bogomips : 4789.45

clflush size : 64

cache\_alignment : 64

address sizes : 46 bits physical, 48 bits virtual

power management:

##### /proc/meminfo #####

MemTotal: 165237704 kB

MemFree: 19945712 kB

Buffers: 1179232 kB

Cached: 129025956 kB

SwapCached: 0 kB

Active: 20598904 kB

Inactive: 119510804 kB

Active(anon): 9977840 kB

Inactive(anon): 176888 kB

Active(file): 10621064 kB

Inactive(file): 119333916 kB

Unevictable: 0 kB

Mlocked: 0 kB

SwapTotal: 0 kB

SwapFree: 0 kB

Dirty: 289028 kB

Writeback: 0 kB

AnonPages: 9885636 kB

Mapped: 456168 kB

Shmem: 269516 kB

Slab: 3992408 kB

SReclaimable: 3920760 kB

SUnreclaim: 71648 kB

KernelStack: 16640 kB

PageTables: 48880 kB

NFS\_Unstable: 0 kB

Bounce: 0 kB

WritebackTmp: 0 kB

CommitLimit: 82618852 kB

Committed\_AS: 27066612 kB

VmallocTotal: 34359738367 kB

VmallocUsed: 427252 kB

VmallocChunk: 34275747196 kB

HardwareCorrupted: 0 kB

AnonHugePages: 489472 kB

HugePages\_Total: 0

HugePages\_Free: 0

HugePages\_Rsvd: 0

HugePages\_Surp: 0

Hugepagesize: 2048 kB

DirectMap4k: 8188 kB

DirectMap2M: 167763968 kB

##### lscpu #####

Architecture: x86\_64

CPU op-mode(s): 32-bit, 64-bit

Byte Order: Little Endian

CPU(s): 40

On-line CPU(s) list: 0-31

Off-line CPU(s) list: 32-39

Thread(s) per core: 1

Core(s) per socket: 10

Socket(s): 2

NUMA node(s): 2

Vendor ID: GenuineIntel

CPU family: 6

Model: 63

Stepping: 2

CPU MHz: 2394.725

BogoMIPS: 4788.60

Hypervisor vendor: Xen

Virtualization type: full

L1d cache: 32K

L1i cache: 32K

L2 cache: 256K

L3 cache: 30720K

NUMA node0 CPU(s): 0-9,20-29

NUMA node1 CPU(s): 10-19,30,31

##### lspci #####

00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)

00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]

00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]

00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 01)

00:02.0 VGA compatible controller: Cirrus Logic GD 5446

00:03.0 Unassigned class [ff80]: XenSource, Inc. Xen Platform Device (rev 01)

##### lsblk #####

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

xvda 202:0 0 250G 0 disk

└─xvda1 202:1 0 250G 0 part /

xvdb 202:16 0 500G 0 disk /hdfs

##### ifconfig #####

eth0 Link encap:Ethernet HWaddr 02:EC:2E:D9:52:AF

inet addr:172.31.40.36 Bcast:172.31.47.255 Mask:255.255.240.0

inet6 addr: fe80::ec:2eff:fed9:52af/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:9001 Metric:1

RX packets:1551206455 errors:0 dropped:0 overruns:0 frame:0

TX packets:913371611 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:7619845808430 (6.9 TiB) TX bytes:8075427723927 (7.3 TiB)

Interrupt:172

lo Link encap:Local Loopback

inet addr:127.0.0.1 Mask:255.0.0.0

inet6 addr: ::1/128 Scope:Host

UP LOOPBACK RUNNING MTU:65536 Metric:1

RX packets:474992213 errors:0 dropped:0 overruns:0 frame:0

TX packets:474992213 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:0

RX bytes:6400983832792 (5.8 TiB) TX bytes:6400983832792 (5.8 TiB)

##############################

# Software #

##############################

##### linux release #####

LSB\_VERSION=base-4.0-amd64:base-4.0-noarch:core-4.0-amd64:core-4.0-noarch:graphics-4.0-amd64:graphics-4.0-noarch:printing-4.0-amd64:printing-4.0-noarch

Red Hat Enterprise Linux Server release 6.6 (Santiago)

Red Hat Enterprise Linux Server release 6.6 (Santiago)

##### kernel release #####

Linux bmarktest01.local.com 2.6.32-504.23.4.el6.x86\_64 #1 SMP Fri May 29 10:16:43 EDT 2015 x86\_64 x86\_64 x86\_64 GNU/Linux

##### date #####

Sat Aug 15 00:14:54 EDT 2015

##### hadoop version #####

Hadoop 2.6.0-cdh5.4.4

Subversion http://github.com/cloudera/hadoop -r b739cd891f6269da5dd22766d7e75bd2c9db73b6

Compiled by jenkins on 2015-07-07T00:02Z

Compiled with protoc 2.5.0

From source with checksum 4acea6ac185376e0b48b33695e88e7a7

This command was run using /opt/cloudera/parcels/CDH-5.4.4-1.cdh5.4.4.p0.4/jars/hadoop-common-2.6.0-cdh5.4.4.jar

##### hadoop classpath #####

/etc/hadoop/conf:/opt/cloudera/parcels/CDH-5.4.4-1.cdh5.4.4.p0.4/lib/hadoop/libexec/../../hadoop/lib/\*:/opt/cloudera/parcels/CDH-5.4.4-1.cdh5.4.4.p0.4/lib/hadoop/libexec/../../hadoop/.//\*:/opt/cloudera/parcels/CDH-5.4.4-1.cdh5.4.4.p0.4/lib/hadoop/libexec/../../hadoop-hdfs/./:/opt/cloudera/parcels/CDH-5.4.4-1.cdh5.4.4.p0.4/lib/hadoop/libexec/../../hadoop-hdfs/lib/\*:/opt/cloudera/parcels/CDH-5.4.4-1.cdh5.4.4.p0.4/lib/hadoop/libexec/../../hadoop-hdfs/.//\*:/opt/cloudera/parcels/CDH-5.4.4-1.cdh5.4.4.p0.4/lib/hadoop/libexec/../../hadoop-yarn/lib/\*:/opt/cloudera/parcels/CDH-5.4.4-1.cdh5.4.4.p0.4/lib/hadoop/libexec/../../hadoop-yarn/.//\*:/opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/lib/\*:/opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/.//\*

##### java version #####

java version "1.7.0\_79"

OpenJDK Runtime Environment (rhel-2.5.5.3.el6\_6-x86\_64 u79-b14)

OpenJDK 64-Bit Server VM (build 24.79-b02, mixed mode)

##### environment #####

BASH=/bin/bash

BASHOPTS=cmdhist:extquote:force\_fignore:hostcomplete:interactive\_comments:progcomp:promptvars:sourcepath

BASH\_ALIASES=()

BASH\_ARGC=([0]="11")

BASH\_ARGV=([0]="-U" [1]="300" [2]="-m" [3]="1000" [4]="-f" [5]="LOAD\_TEST" [6]="-i" [7]="1000" [8]="-f" [9]="-b" [10]="zipQueryLogs")

BASH\_CMDS=()

BASH\_LINENO=([0]="465" [1]="0")

BASH\_SOURCE=([0]="/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/bin/bigBench" [1]="/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/bin/bigBench")

BASH\_VERSINFO=([0]="4" [1]="1" [2]="2" [3]="1" [4]="release" [5]="x86\_64-redhat-linux-gnu")

BASH\_VERSION='4.1.2(1)-release'

BIG\_BENCH\_BENCHMARK\_PHASE=run\_Query

BIG\_BENCH\_BIN\_DIR=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/bin

BIG\_BENCH\_CLEAN\_DIR=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/engines/hive/clean

BIG\_BENCH\_CLEAN\_METASTORE\_FILE=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/engines/hive/clean/dropTables.sql

BIG\_BENCH\_CONF\_DIR=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/conf

BIG\_BENCH\_DATABASE=bigbenchORC

BIG\_BENCH\_DATAGEN\_CORE\_SITE=/etc/hadoop/conf.cloudera.hdfs/core-site.xml

BIG\_BENCH\_DATAGEN\_DFS\_REPLICATION=3

BIG\_BENCH\_DATAGEN\_HADOOP\_EXEC\_DEBUG=

BIG\_BENCH\_DATAGEN\_HADOOP\_JVM\_ENV='java -Xmx800m'

BIG\_BENCH\_DATAGEN\_HADOOP\_OPTIONS=' -workers 1 -ap 3000 '

BIG\_BENCH\_DATAGEN\_HDFS\_SITE=/etc/hadoop/conf.cloudera.hdfs/hdfs-site.xml

BIG\_BENCH\_DATAGEN\_STAGE\_LOG=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/logs/dataGeneration-run\_Query.log

BIG\_BENCH\_DATAGEN\_TABLES=

BIG\_BENCH\_DATA\_GENERATOR\_DIR=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/data-generator

BIG\_BENCH\_DEFAULT\_BENCHMARK\_PHASE=run\_Query

BIG\_BENCH\_DEFAULT\_DATABASE=bigbenchORC

BIG\_BENCH\_DEFAULT\_ENGINE=hive

BIG\_BENCH\_DEFAULT\_MAP\_TASKS=80

BIG\_BENCH\_DEFAULT\_NUMBER\_OF\_PARALLEL\_STREAMS=2

BIG\_BENCH\_DEFAULT\_SCALE\_FACTOR=10

BIG\_BENCH\_ENGINE=hive

BIG\_BENCH\_ENGINE\_BIN\_DIR=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/engines/hive/bin

BIG\_BENCH\_ENGINE\_CONF\_DIR=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/engines/hive/conf

BIG\_BENCH\_ENGINE\_DIR=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/engines/hive

BIG\_BENCH\_ENGINE\_HIVE\_MAHOUT\_EXECUTION=sequential

BIG\_BENCH\_ENGINE\_SETTINGS\_FILE=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/engines/hive/conf/hiveSettings.sql

BIG\_BENCH\_EXPERT\_MODE=1

BIG\_BENCH\_HADOOP\_CONF=/etc/hadoop/conf.cloudera.hdfs

BIG\_BENCH\_HADOOP\_LIBS\_NATIVE=/opt/cloudera/parcels/CDH/lib/hadoop/lib/native

BIG\_BENCH\_HDFS\_ABSOLUTE\_HOME=/user/ec2-user/benchmarks/bigbench

BIG\_BENCH\_HDFS\_ABSOLUTE\_INIT\_DATA\_DIR=/user/ec2-user/benchmarks/bigbench/data

BIG\_BENCH\_HDFS\_ABSOLUTE\_PATH=/user/ec2-user

BIG\_BENCH\_HDFS\_ABSOLUTE\_QUERY\_RESULT\_DIR=/user/ec2-user/benchmarks/bigbench/queryResults

BIG\_BENCH\_HDFS\_ABSOLUTE\_REFRESH\_DATA\_DIR=/user/ec2-user/benchmarks/bigbench/data\_refresh

BIG\_BENCH\_HDFS\_ABSOLUTE\_TEMP\_DIR=/user/ec2-user/benchmarks/bigbench/temp

BIG\_BENCH\_HDFS\_RELATIVE\_HOME=benchmarks/bigbench

BIG\_BENCH\_HDFS\_RELATIVE\_INIT\_DATA\_DIR=benchmarks/bigbench/data

BIG\_BENCH\_HDFS\_RELATIVE\_QUERY\_RESULT\_DIR=benchmarks/bigbench/queryResults

BIG\_BENCH\_HDFS\_RELATIVE\_REFRESH\_DATA\_DIR=benchmarks/bigbench/data\_refresh

BIG\_BENCH\_HDFS\_RELATIVE\_TEMP\_DIR=benchmarks/bigbench/temp

BIG\_BENCH\_HOME=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans

BIG\_BENCH\_JAVA=java

BIG\_BENCH\_LOADING\_STAGE\_LOG=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/logs/populateMetastore-run\_Query.log

BIG\_BENCH\_LOGS\_DIR=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/logs

BIG\_BENCH\_MAP\_TASKS=300

BIG\_BENCH\_NUMBER\_OF\_PARALLEL\_STREAMS=2

BIG\_BENCH\_POPULATE\_METASTORE\_FILE=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/engines/hive/population/hiveCreateLoad\_decimal.sql

BIG\_BENCH\_POPULATION\_DIR=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/engines/hive/population

BIG\_BENCH\_QUERIES\_DIR=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/engines/hive/**Queries**

BIG\_BENCH\_QUERY\_PARAMS\_FILE=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/engines/hive/conf/queryParameters.sql

BIG\_BENCH\_REFRESH\_DIR=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/engines/hive/refresh

BIG\_BENCH\_REFRESH\_METASTORE\_FILE=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/engines/hive/refresh/hiveRefreshCreateLoad\_decimal.sql

BIG\_BENCH\_SCALE\_FACTOR=1000

BIG\_BENCH\_STOP\_AFTER\_FAILURE=0

BIG\_BENCH\_STREAM\_NUMBER=0

BIG\_BENCH\_TOOLS\_DIR=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/tools

BIG\_BENCH\_USER=ec2-user

BIG\_BENCH\_USE\_SNAKEBITE\_HDFSCLIENT=0

BIG\_BENCH\_hive\_default\_fileformat\_result\_table=TEXTFILE

BIG\_BENCH\_hive\_default\_fileformat\_source\_table=ORC

BIG\_BENCH\_java\_child\_process\_xmx=' -Xmx1024m '

BINARY=/usr/bin/hive

BINARY\_PARAMS=()

CVS\_RSH=ssh

DIRSTACK=()

ENGINE\_RUN\_METHOD=runEngineCmd

ENGINE\_SETTINGS=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/engines/hive/conf/engineSettings.conf

ENV\_INFO\_FILE=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/logs/envInfo.log

EUID=500

FUNCNAME=([0]="logEnvInformation" [1]="main")

GROUPS=()

G\_BROKEN\_FILENAMES=1

HISTCONTROL=ignoredups

HISTSIZE=1000

HOME=/home/ec2-user

HOSTNAME=bmarktest01.local.com

HOSTTYPE=x86\_64

IFS=$' \t\n'

JAVA\_HOME=/usr/lib/jvm/jre

LANG=en\_US.UTF-8

LESSOPEN='||/usr/bin/lesspipe.sh %s'

LIST\_OF\_USER\_OPTIONS='-b -f 1000 -i LOAD\_TEST -f 1000 -m 300 -U'

LOGNAME=ec2-user

LS\_COLORS='rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=01;05;37;41:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:\*.tar=01;31:\*.tgz=01;31:\*.arj=01;31:\*.taz=01;31:\*.lzh=01;31:\*.lzma=01;31:\*.tlz=01;31:\*.txz=01;31:\*.zip=01;31:\*.z=01;31:\*.Z=01;31:\*.dz=01;31:\*.gz=01;31:\*.lz=01;31:\*.xz=01;31:\*.bz2=01;31:\*.tbz=01;31:\*.tbz2=01;31:\*.bz=01;31:\*.tz=01;31:\*.deb=01;31:\*.rpm=01;31:\*.jar=01;31:\*.rar=01;31:\*.ace=01;31:\*.zoo=01;31:\*.cpio=01;31:\*.7z=01;31:\*.rz=01;31:\*.jpg=01;35:\*.jpeg=01;35:\*.gif=01;35:\*.bmp=01;35:\*.pbm=01;35:\*.pgm=01;35:\*.ppm=01;35:\*.tga=01;35:\*.xbm=01;35:\*.xpm=01;35:\*.tif=01;35:\*.tiff=01;35:\*.png=01;35:\*.svg=01;35:\*.svgz=01;35:\*.mng=01;35:\*.pcx=01;35:\*.mov=01;35:\*.mpg=01;35:\*.mpeg=01;35:\*.m2v=01;35:\*.mkv=01;35:\*.ogm=01;35:\*.mp4=01;35:\*.m4v=01;35:\*.mp4v=01;35:\*.vob=01;35:\*.qt=01;35:\*.nuv=01;35:\*.wmv=01;35:\*.asf=01;35:\*.rm=01;35:\*.rmvb=01;35:\*.flc=01;35:\*.avi=01;35:\*.fli=01;35:\*.flv=01;35:\*.gl=01;35:\*.dl=01;35:\*.xcf=01;35:\*.xwd=01;35:\*.yuv=01;35:\*.cgm=01;35:\*.emf=01;35:\*.axv=01;35:\*.anx=01;35:\*.ogv=01;35:\*.ogx=01;35:\*.aac=01;36:\*.au=01;36:\*.flac=01;36:\*.mid=01;36:\*.midi=01;36:\*.mka=01;36:\*.mp3=01;36:\*.mpc=01;36:\*.ogg=01;36:\*.ra=01;36:\*.wav=01;36:\*.axa=01;36:\*.oga=01;36:\*.spx=01;36:\*.xspf=01;36:'

MACHTYPE=x86\_64-redhat-linux-gnu

MAIL=/var/spool/mail/ec2-user

MODULE=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/bin/zipQueryLogs

MODULE\_HELP\_METHOD=helpModule

MODULE\_NAME=zipQueryLogs

MODULE\_RUN\_METHOD=runModule

NLSPATH=/usr/dt/lib/nls/msg/%L/%N.cat

OLDPWD=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans

OPT='?'

OPTERR=1

OPTIND=11

OSTYPE=linux-gnu

PATH=/usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/ec2-user/bin

PIPESTATUS=([0]="0")

PPID=63520

PS4='+ '

PWD=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans

QTDIR=/usr/lib64/qt-3.3

QTINC=/usr/lib64/qt-3.3/include

QTLIB=/usr/lib64/qt-3.3/lib

SHELL=/bin/bash

SHELLOPTS=braceexpand:hashall:interactive-comments

SHLVL=4

SHOW\_HELP=0

SSH\_CLIENT='172.31.22.134 58617 22'

SSH\_CONNECTION='172.31.22.134 58617 172.31.40.36 22'

SSH\_TTY=/dev/pts/1

TERM=xterm

UID=500

USER=ec2-user

USER\_DRIVER\_WORKLOAD=LOAD\_TEST

USER\_EXPERT\_MODE=1

USER\_MAP\_TASKS=300

USER\_PRINT\_STD\_OUT=1

USER\_SCALE\_FACTOR=1000

USER\_SETTINGS=/home/ec2-user/Big-Data-Benchmark-for-Big-Bench-MasterVersion\_14\_Aug\_incl\_kmeans/conf/userSettings.conf

XFILESEARCHPATH=/usr/dt/app-defaults/%L/Dt

1. – Data Redundancy Report

Sample output of command: hdfs fsck -blocks

FSCK started by root (auth:SIMPLE) from /10.54.6.153 for path / at Fri Aug 09 00:17:10 PDT 2019

Status: HEALTHY

Number of data-nodes: 7

Number of racks: 1

Total dirs: 4258

Total symlinks: 0

Replicated Blocks:

Total size: 6012044889004 B

Total files: 136593

Total blocks (validated): 121145 (avg. block size 49626851 B)

Minimally replicated blocks: 121145 (100.0 %)

Over-replicated blocks: 0 (0.0 %)

Under-replicated blocks: 0 (0.0 %)

Mis-replicated blocks: 0 (0.0 %)

Default replication factor: 3

Average block replication: 3.0000165

Missing blocks: 0

Corrupt blocks: 0

Missing replicas: 0 (0.0 %)

Erasure Coded Block Groups:

Total size: 0 B

Total files: 0

Total block groups (validated): 0

Minimally erasure-coded block groups: 0

Over-erasure-coded block groups: 0

Under-erasure-coded block groups: 0

Unsatisfactory placement block groups: 0

Average block group size: 0.0

Missing block groups: 0

Corrupt block groups: 0

Missing internal blocks: 0

FSCK ended at Fri Aug 09 00:17:11 PDT 2019 in 1618 milliseconds

The filesystem under path '/' is HEALTHY

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Sample output of command: hdfs dfs -du -s -h benchmarks/bigbench/data

1.0 G 3.1 G benchmarks/bigbench/data

1. – Custom Load Script

set hdfsDataPath=${env:BIG\_BENCH\_HDFS\_ABSOLUTE\_INIT\_DATA\_DIR};

set fieldDelimiter=|;

set tableFormat=${env:BIG\_BENCH\_hive\_default\_fileformat\_source\_table};

set temporaryTableSuffix=\_temporary;

set customerTableName=customer;

set customerAddressTableName=customer\_address;

set customerDemographicsTableName=customer\_demographics;

set dateTableName=date\_dim;

set householdDemographicsTableName=household\_demographics;

set incomeTableName=income\_band;

set itemTableName=item;

set promotionTableName=promotion;

set reasonTableName=reason;

set shipModeTableName=ship\_mode;

set storeTableName=store;

set timeTableName=time\_dim;

set warehouseTableName=warehouse;

set webSiteTableName=web\_site;

set webPageTableName=web\_page;

set inventoryTableName=inventory;

set storeSalesTableName=store\_sales;

set storeReturnsTableName=store\_returns;

set webSalesTableName=web\_sales;

set webReturnsTableName=web\_returns;

set marketPricesTableName=item\_marketprices;

set clickstreamsTableName=web\_clickstreams;

set reviewsTableName=product\_reviews;

-- !echo Create temporary table: ${hiveconf:customerTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:customerTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:customerTableName}${hiveconf:temporaryTableSuffix}

( c\_customer\_sk bigint --not null

, c\_customer\_id string --not null

, c\_current\_cdemo\_sk bigint

, c\_current\_hdemo\_sk bigint

, c\_current\_addr\_sk bigint

, c\_first\_shipto\_date\_sk bigint

, c\_first\_sales\_date\_sk bigint

, c\_salutation string

, c\_first\_name string

, c\_last\_name string

, c\_preferred\_cust\_flag string

, c\_birth\_day int

, c\_birth\_month int

, c\_birth\_year int

, c\_birth\_country string

, c\_login string

, c\_email\_address string

, c\_last\_review\_date string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:customerTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:customerTableName};

DROP TABLE IF EXISTS ${hiveconf:customerTableName};

CREATE TABLE ${hiveconf:customerTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:customerTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:customerTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:customerTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:customerAddressTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:customerAddressTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:customerAddressTableName}${hiveconf:temporaryTableSuffix}

( ca\_address\_sk bigint --not null

, ca\_address\_id string --not null

, ca\_street\_number string

, ca\_street\_name string

, ca\_street\_type string

, ca\_suite\_number string

, ca\_city string

, ca\_county string

, ca\_state string

, ca\_zip string

, ca\_country string

, ca\_gmt\_offset decimal(5,2)

, ca\_location\_type string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:customerAddressTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:customerAddressTableName};

DROP TABLE IF EXISTS ${hiveconf:customerAddressTableName};

CREATE TABLE ${hiveconf:customerAddressTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:customerAddressTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:customerAddressTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:customerAddressTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:customerDemographicsTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:customerDemographicsTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:customerDemographicsTableName}${hiveconf:temporaryTableSuffix}

( cd\_demo\_sk bigint ----not null

, cd\_gender string

, cd\_marital\_status string

, cd\_education\_status string

, cd\_purchase\_estimate int

, cd\_credit\_rating string

, cd\_dep\_count int

, cd\_dep\_employed\_count int

, cd\_dep\_college\_count int

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:customerDemographicsTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:customerDemographicsTableName};

DROP TABLE IF EXISTS ${hiveconf:customerDemographicsTableName};

CREATE TABLE ${hiveconf:customerDemographicsTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:customerDemographicsTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:customerDemographicsTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:customerDemographicsTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:dateTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:dateTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:dateTableName}${hiveconf:temporaryTableSuffix}

( d\_date\_sk bigint --not null

, d\_date\_id string --not null

, d\_date string

, d\_month\_seq int

, d\_week\_seq int

, d\_quarter\_seq int

, d\_year int

, d\_dow int

, d\_moy int

, d\_dom int

, d\_qoy int

, d\_fy\_year int

, d\_fy\_quarter\_seq int

, d\_fy\_week\_seq int

, d\_day\_name string

, d\_quarter\_name string

, d\_holiday string

, d\_weekend string

, d\_following\_holiday string

, d\_first\_dom int

, d\_last\_dom int

, d\_same\_day\_ly int

, d\_same\_day\_lq int

, d\_current\_day string

, d\_current\_week string

, d\_current\_month string

, d\_current\_quarter string

, d\_current\_year string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:dateTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:dateTableName};

DROP TABLE IF EXISTS ${hiveconf:dateTableName};

CREATE TABLE ${hiveconf:dateTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:dateTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:dateTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:dateTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:householdDemographicsTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:householdDemographicsTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:householdDemographicsTableName}${hiveconf:temporaryTableSuffix}

( hd\_demo\_sk bigint --not null

, hd\_income\_band\_sk bigint

, hd\_buy\_potential string

, hd\_dep\_count int

, hd\_vehicle\_count int

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:householdDemographicsTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:householdDemographicsTableName};

DROP TABLE IF EXISTS ${hiveconf:householdDemographicsTableName};

CREATE TABLE ${hiveconf:householdDemographicsTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:householdDemographicsTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:householdDemographicsTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:householdDemographicsTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:incomeTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:incomeTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:incomeTableName}${hiveconf:temporaryTableSuffix}

( ib\_income\_band\_sk bigint --not null

, ib\_lower\_bound int

, ib\_upper\_bound int

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:incomeTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:incomeTableName};

DROP TABLE IF EXISTS ${hiveconf:incomeTableName};

CREATE TABLE ${hiveconf:incomeTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:incomeTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:incomeTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:incomeTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:itemTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:itemTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:itemTableName}${hiveconf:temporaryTableSuffix}

( i\_item\_sk bigint --not null

, i\_item\_id string --not null

, i\_rec\_start\_date string

, i\_rec\_end\_date string

, i\_item\_desc string

, i\_current\_price decimal(7,2)

, i\_wholesale\_cost decimal(7,2)

, i\_brand\_id int

, i\_brand string

, i\_class\_id int

, i\_class string

, i\_category\_id int

, i\_category string

, i\_manufact\_id int

, i\_manufact string

, i\_size string

, i\_formulation string

, i\_color string

, i\_units string

, i\_container string

, i\_manager\_id int

, i\_product\_name string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:itemTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:itemTableName};

DROP TABLE IF EXISTS ${hiveconf:itemTableName};

CREATE TABLE ${hiveconf:itemTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:itemTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:itemTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:itemTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:promotionTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:promotionTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:promotionTableName}${hiveconf:temporaryTableSuffix}

( p\_promo\_sk bigint --not null

, p\_promo\_id string --not null

, p\_start\_date\_sk bigint

, p\_end\_date\_sk bigint

, p\_item\_sk bigint

, p\_cost decimal(15,2)

, p\_response\_target int

, p\_promo\_name string

, p\_channel\_dmail string

, p\_channel\_email string

, p\_channel\_catalog string

, p\_channel\_tv string

, p\_channel\_radio string

, p\_channel\_press string

, p\_channel\_event string

, p\_channel\_demo string

, p\_channel\_details string

, p\_purpose string

, p\_discount\_active string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:promotionTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:promotionTableName};

DROP TABLE IF EXISTS ${hiveconf:promotionTableName};

CREATE TABLE ${hiveconf:promotionTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:promotionTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:promotionTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:promotionTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:reasonTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:reasonTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:reasonTableName}${hiveconf:temporaryTableSuffix}

( r\_reason\_sk bigint --not null

, r\_reason\_id string --not null

, r\_reason\_desc string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:reasonTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:reasonTableName};

DROP TABLE IF EXISTS ${hiveconf:reasonTableName};

CREATE TABLE ${hiveconf:reasonTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:reasonTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:reasonTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:reasonTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:shipModeTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:shipModeTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:shipModeTableName}${hiveconf:temporaryTableSuffix}

( sm\_ship\_mode\_sk bigint --not null

, sm\_ship\_mode\_id string --not null

, sm\_type string

, sm\_code string

, sm\_carrier string

, sm\_contract string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:shipModeTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:shipModeTableName};

DROP TABLE IF EXISTS ${hiveconf:shipModeTableName};

CREATE TABLE ${hiveconf:shipModeTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:shipModeTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:shipModeTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:shipModeTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:storeTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:storeTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:storeTableName}${hiveconf:temporaryTableSuffix}

( s\_store\_sk bigint --not null

, s\_store\_id string --not null

, s\_rec\_start\_date string

, s\_rec\_end\_date string

, s\_closed\_date\_sk bigint

, s\_store\_name string

, s\_number\_employees int

, s\_floor\_space int

, s\_hours string

, s\_manager string

, s\_market\_id int

, s\_geography\_class string

, s\_market\_desc string

, s\_market\_manager string

, s\_division\_id int

, s\_division\_name string

, s\_company\_id int

, s\_company\_name string

, s\_street\_number string

, s\_street\_name string

, s\_street\_type string

, s\_suite\_number string

, s\_city string

, s\_county string

, s\_state string

, s\_zip string

, s\_country string

, s\_gmt\_offset decimal(5,2)

, s\_tax\_precentage decimal(5,2)

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:storeTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:storeTableName};

DROP TABLE IF EXISTS ${hiveconf:storeTableName};

CREATE TABLE ${hiveconf:storeTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:storeTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:storeTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:storeTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:timeTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:timeTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:timeTableName}${hiveconf:temporaryTableSuffix}

( t\_time\_sk bigint --not null

, t\_time\_id string --not null

, t\_time int

, t\_hour int

, t\_minute int

, t\_second int

, t\_am\_pm string

, t\_shift string

, t\_sub\_shift string

, t\_meal\_time string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:timeTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:timeTableName};

DROP TABLE IF EXISTS ${hiveconf:timeTableName};

CREATE TABLE ${hiveconf:timeTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:timeTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:timeTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:timeTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:warehouseTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:warehouseTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:warehouseTableName}${hiveconf:temporaryTableSuffix}

( w\_warehouse\_sk bigint --not null

, w\_warehouse\_id string --not null

, w\_warehouse\_name string

, w\_warehouse\_sq\_ft int

, w\_street\_number string

, w\_street\_name string

, w\_street\_type string

, w\_suite\_number string

, w\_city string

, w\_county string

, w\_state string

, w\_zip string

, w\_country string

, w\_gmt\_offset decimal(5,2)

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:warehouseTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:warehouseTableName};

DROP TABLE IF EXISTS ${hiveconf:warehouseTableName};

CREATE TABLE ${hiveconf:warehouseTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:warehouseTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:warehouseTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:warehouseTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:webSiteTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:webSiteTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:webSiteTableName}${hiveconf:temporaryTableSuffix}

( web\_site\_sk bigint --not null

, web\_site\_id string --not null

, web\_rec\_start\_date string

, web\_rec\_end\_date string

, web\_name string

, web\_open\_date\_sk bigint

, web\_close\_date\_sk bigint

, web\_class string

, web\_manager string

, web\_mkt\_id int

, web\_mkt\_class string

, web\_mkt\_desc string

, web\_market\_manager string

, web\_company\_id int

, web\_company\_name string

, web\_street\_number string

, web\_street\_name string

, web\_street\_type string

, web\_suite\_number string

, web\_city string

, web\_county string

, web\_state string

, web\_zip string

, web\_country string

, web\_gmt\_offset decimal(5,2)

, web\_tax\_percentage decimal(5,2)

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:webSiteTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:webSiteTableName};

DROP TABLE IF EXISTS ${hiveconf:webSiteTableName};

CREATE TABLE ${hiveconf:webSiteTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:webSiteTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:webSiteTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:webSiteTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:webPageTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:webPageTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:webPageTableName}${hiveconf:temporaryTableSuffix}

( wp\_web\_page\_sk bigint --not null

, wp\_web\_page\_id string --not null

, wp\_rec\_start\_date string

, wp\_rec\_end\_date string

, wp\_creation\_date\_sk bigint

, wp\_access\_date\_sk bigint

, wp\_autogen\_flag string

, wp\_customer\_sk bigint

, wp\_url string

, wp\_type string

, wp\_char\_count int

, wp\_link\_count int

, wp\_image\_count int

, wp\_max\_ad\_count int

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:webPageTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:webPageTableName};

DROP TABLE IF EXISTS ${hiveconf:webPageTableName};

CREATE TABLE ${hiveconf:webPageTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:webPageTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:webPageTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:webPageTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:inventoryTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:inventoryTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:inventoryTableName}${hiveconf:temporaryTableSuffix}

( inv\_date\_sk bigint --not null

, inv\_item\_sk bigint --not null

, inv\_warehouse\_sk bigint --not null

, inv\_quantity\_on\_hand int

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:inventoryTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:inventoryTableName};

DROP TABLE IF EXISTS ${hiveconf:inventoryTableName};

CREATE TABLE ${hiveconf:inventoryTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:inventoryTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:inventoryTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:inventoryTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:storeSalesTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:storeSalesTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:storeSalesTableName}${hiveconf:temporaryTableSuffix}

( ss\_sold\_date\_sk bigint

, ss\_sold\_time\_sk bigint

, ss\_item\_sk bigint --not null

, ss\_customer\_sk bigint

, ss\_cdemo\_sk bigint

, ss\_hdemo\_sk bigint

, ss\_addr\_sk bigint

, ss\_store\_sk bigint

, ss\_promo\_sk bigint

, ss\_ticket\_number bigint --not null

, ss\_quantity int

, ss\_wholesale\_cost decimal(7,2)

, ss\_list\_price decimal(7,2)

, ss\_sales\_price decimal(7,2)

, ss\_ext\_discount\_amt decimal(7,2)

, ss\_ext\_sales\_price decimal(7,2)

, ss\_ext\_wholesale\_cost decimal(7,2)

, ss\_ext\_list\_price decimal(7,2)

, ss\_ext\_tax decimal(7,2)

, ss\_coupon\_amt decimal(7,2)

, ss\_net\_paid decimal(7,2)

, ss\_net\_paid\_inc\_tax decimal(7,2)

, ss\_net\_profit decimal(7,2)

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:storeSalesTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:storeSalesTableName};

DROP TABLE IF EXISTS ${hiveconf:storeSalesTableName};

CREATE TABLE ${hiveconf:storeSalesTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:storeSalesTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:storeSalesTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:storeSalesTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:storeReturnsTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:storeReturnsTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:storeReturnsTableName}${hiveconf:temporaryTableSuffix}

( sr\_returned\_date\_sk bigint

, sr\_return\_time\_sk bigint

, sr\_item\_sk bigint --not null

, sr\_customer\_sk bigint

, sr\_cdemo\_sk bigint

, sr\_hdemo\_sk bigint

, sr\_addr\_sk bigint

, sr\_store\_sk bigint

, sr\_reason\_sk bigint

, sr\_ticket\_number bigint --not null

, sr\_return\_quantity int

, sr\_return\_amt decimal(7,2)

, sr\_return\_tax decimal(7,2)

, sr\_return\_amt\_inc\_tax decimal(7,2)

, sr\_fee decimal(7,2)

, sr\_return\_ship\_cost decimal(7,2)

, sr\_refunded\_cash decimal(7,2)

, sr\_reversed\_charge decimal(7,2)

, sr\_store\_credit decimal(7,2)

, sr\_net\_loss decimal(7,2)

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:storeReturnsTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:storeReturnsTableName};

DROP TABLE IF EXISTS ${hiveconf:storeReturnsTableName};

CREATE TABLE ${hiveconf:storeReturnsTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:storeReturnsTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:storeReturnsTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:storeReturnsTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:webSalesTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:webSalesTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:webSalesTableName}${hiveconf:temporaryTableSuffix}

( ws\_sold\_date\_sk bigint

, ws\_sold\_time\_sk bigint

, ws\_ship\_date\_sk bigint

, ws\_item\_sk bigint --not null

, ws\_bill\_customer\_sk bigint

, ws\_bill\_cdemo\_sk bigint

, ws\_bill\_hdemo\_sk bigint

, ws\_bill\_addr\_sk bigint

, ws\_ship\_customer\_sk bigint

, ws\_ship\_cdemo\_sk bigint

, ws\_ship\_hdemo\_sk bigint

, ws\_ship\_addr\_sk bigint

, ws\_web\_page\_sk bigint

, ws\_web\_site\_sk bigint

, ws\_ship\_mode\_sk bigint

, ws\_warehouse\_sk bigint

, ws\_promo\_sk bigint

, ws\_order\_number bigint --not null

, ws\_quantity int

, ws\_wholesale\_cost decimal(7,2)

, ws\_list\_price decimal(7,2)

, ws\_sales\_price decimal(7,2)

, ws\_ext\_discount\_amt decimal(7,2)

, ws\_ext\_sales\_price decimal(7,2)

, ws\_ext\_wholesale\_cost decimal(7,2)

, ws\_ext\_list\_price decimal(7,2)

, ws\_ext\_tax decimal(7,2)

, ws\_coupon\_amt decimal(7,2)

, ws\_ext\_ship\_cost decimal(7,2)

, ws\_net\_paid decimal(7,2)

, ws\_net\_paid\_inc\_tax decimal(7,2)

, ws\_net\_paid\_inc\_ship decimal(7,2)

, ws\_net\_paid\_inc\_ship\_tax decimal(7,2)

, ws\_net\_profit decimal(7,2)

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:webSalesTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:webSalesTableName};

DROP TABLE IF EXISTS ${hiveconf:webSalesTableName};

CREATE TABLE ${hiveconf:webSalesTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:webSalesTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:webSalesTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:webSalesTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:webReturnsTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:webReturnsTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:webReturnsTableName}${hiveconf:temporaryTableSuffix}

( wr\_returned\_date\_sk bigint

, wr\_returned\_time\_sk bigint

, wr\_item\_sk bigint --not null

, wr\_refunded\_customer\_sk bigint

, wr\_refunded\_cdemo\_sk bigint

, wr\_refunded\_hdemo\_sk bigint

, wr\_refunded\_addr\_sk bigint

, wr\_returning\_customer\_sk bigint

, wr\_returning\_cdemo\_sk bigint

, wr\_returning\_hdemo\_sk bigint

, wr\_returning\_addr\_sk bigint

, wr\_web\_page\_sk bigint

, wr\_reason\_sk bigint

, wr\_order\_number bigint --not null

, wr\_return\_quantity int

, wr\_return\_amt decimal(7,2)

, wr\_return\_tax decimal(7,2)

, wr\_return\_amt\_inc\_tax decimal(7,2)

, wr\_fee decimal(7,2)

, wr\_return\_ship\_cost decimal(7,2)

, wr\_refunded\_cash decimal(7,2)

, wr\_reversed\_charge decimal(7,2)

, wr\_account\_credit decimal(7,2)

, wr\_net\_loss decimal(7,2)

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:webReturnsTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:webReturnsTableName};

DROP TABLE IF EXISTS ${hiveconf:webReturnsTableName};

CREATE TABLE ${hiveconf:webReturnsTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:webReturnsTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:webReturnsTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:webReturnsTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:marketPricesTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:marketPricesTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:marketPricesTableName}${hiveconf:temporaryTableSuffix}

( imp\_sk bigint --not null

, imp\_item\_sk bigint --not null

, imp\_competitor string

, imp\_competitor\_price decimal(7,2)

, imp\_start\_date bigint

, imp\_end\_date bigint

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:marketPricesTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:marketPricesTableName};

DROP TABLE IF EXISTS ${hiveconf:marketPricesTableName};

CREATE TABLE ${hiveconf:marketPricesTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:marketPricesTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:marketPricesTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:marketPricesTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:clickstreamsTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:clickstreamsTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:clickstreamsTableName}${hiveconf:temporaryTableSuffix}

( wcs\_click\_date\_sk bigint

, wcs\_click\_time\_sk bigint

, wcs\_sales\_sk bigint

, wcs\_item\_sk bigint

, wcs\_web\_page\_sk bigint

, wcs\_user\_sk bigint

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:clickstreamsTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:clickstreamsTableName};

DROP TABLE IF EXISTS ${hiveconf:clickstreamsTableName};

CREATE TABLE ${hiveconf:clickstreamsTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:clickstreamsTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:clickstreamsTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:clickstreamsTableName}${hiveconf:temporaryTableSuffix};

-- !echo Create temporary table: ${hiveconf:reviewsTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE IF EXISTS ${hiveconf:reviewsTableName}${hiveconf:temporaryTableSuffix};

CREATE EXTERNAL TABLE ${hiveconf:reviewsTableName}${hiveconf:temporaryTableSuffix}

( pr\_review\_sk bigint --not null

, pr\_review\_date string

, pr\_review\_time string

, pr\_review\_rating int --not null

, pr\_item\_sk bigint --not null

, pr\_user\_sk bigint

, pr\_order\_sk bigint

, pr\_review\_content string --not null

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '${hiveconf:fieldDelimiter}'

STORED AS TEXTFILE LOCATION '${hiveconf:hdfsDataPath}/${hiveconf:reviewsTableName}'

;

-- !echo Load text data into ${hiveconf:tableFormat} table: ${hiveconf:reviewsTableName};

DROP TABLE IF EXISTS ${hiveconf:reviewsTableName};

CREATE TABLE ${hiveconf:reviewsTableName}

STORED AS ${hiveconf:tableFormat}

AS

SELECT \* FROM ${hiveconf:reviewsTableName}${hiveconf:temporaryTableSuffix}

;

-- !echo Drop temporary table: ${hiveconf:reviewsTableName}${hiveconf:temporaryTableSuffix};

DROP TABLE ${hiveconf:reviewsTableName}${hiveconf:temporaryTableSuffix};

1. – Throughput Test Stream Placement





1. <https://docs.cloudera.com/documentation/enterprise/6/6.3/topics/cm_mc_dn.html> [↑](#footnote-ref-1)
2. <https://docs.oracle.com/cd/A91202_01/901_doc/rac.901/a89867/pshavdtl.htm> [↑](#footnote-ref-2)
3. <https://www.dummies.com/programming/big-data/hadoop/master-nodes-in-hadoop-clusters/> [↑](#footnote-ref-3)
4. <https://docs.cloudera.com/documentation/enterprise/6/6.3/topics/cm_mc_nn.html> [↑](#footnote-ref-4)