TPC Benchmark™ H
Full Disclosure Report

CPI Phoenix IQ-201 using EXASolution 2.0

First Edition
April 2, 2008

First Edition – April 2, 2008

CPI Phoenix IQ-201
using
EXASolution 2.0

CPI Computer Partner Handels GmbH and EXASOL AG believe that the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. We assume no responsibility for any errors that may appear in this document. The pricing information in this document is believed to accurately reflect the current prices as of the publication date. However, we provide no warranty of the pricing information in this document.

Benchmark results are highly dependent upon workload, specific application requirements, system design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC Benchmark™ H should not be used as a substitute for a specific customer application benchmark when critical capacity planning and/or product evaluation decisions are contemplated.

All performance data contained in this report were obtained in a rigorously controlled environment. Results obtained in other operating environments may vary significantly. We do not warrant or represent that a user can or will achieve similar performance. No warranty of system performance or price/performance is expressed or implied in this report.

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark H test conducted with the EXASolution 2.0 based on a CPI Phoenix IQ-201 Server Cluster in conformance with the requirements of the TPC-H Benchmark Specification.

The benchmark results are summarized in the following table.

Software	Hardware	Total System Cost	QphH @300GB	Price/QphH @300GB
EXASolution 2.0	**CPI Phoenix IQ-201**	**$ 574,565**	**547,205.4**	**$ 1.05**

The Transaction Processing Performance Council (TPC) developed the TPC-H Benchmark. The TPC was founded to define transactions processing benchmarks and to disseminate objective, verifiable performance data to the industry.

Standard and Executive Summary Statements

The Executive Summary and Numerical Quantities Summary of the benchmark results for the CPI Phoenix IQ-201 Cluster can be found in the following pages.

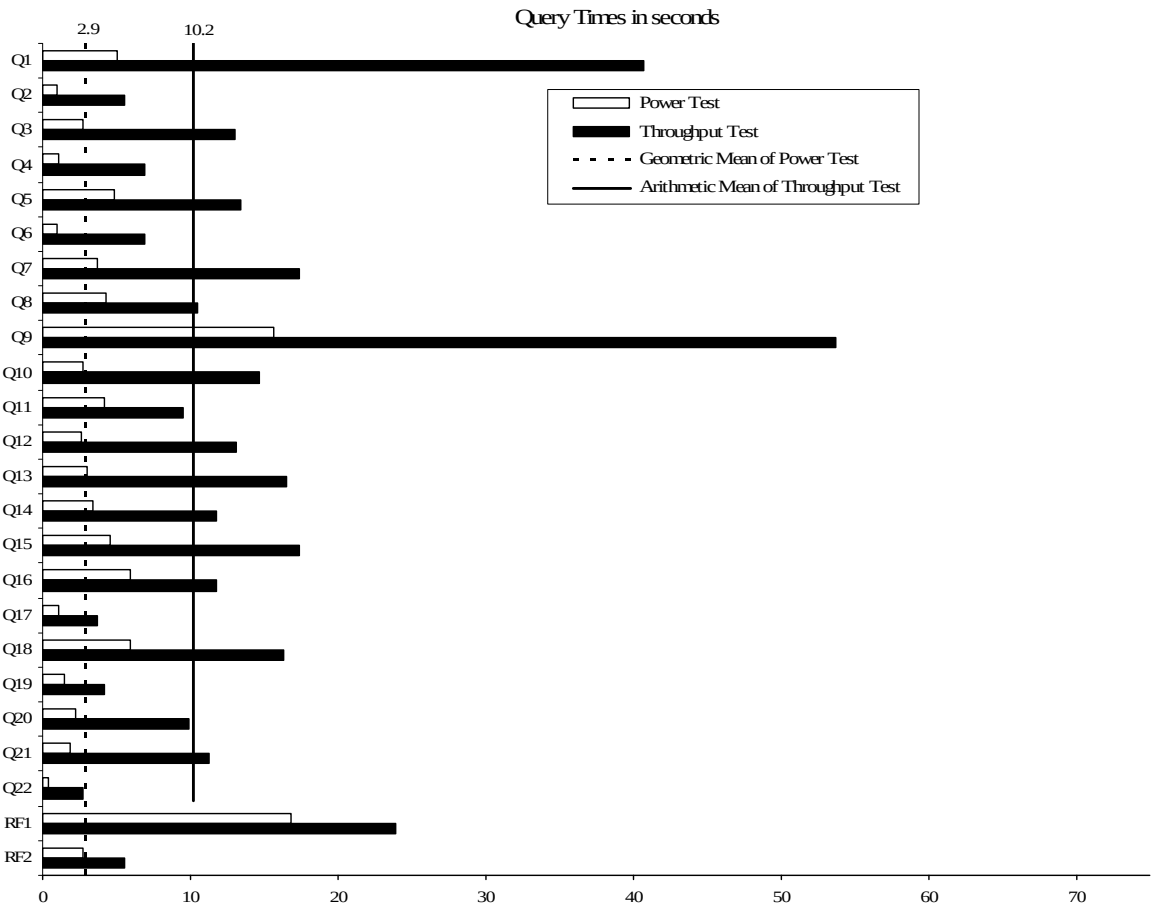
Auditor

In order to verify compliance to the TPC-H benchmark specification, Francois Raab, Infosizing, Inc., audited the benchmark configuration, environment, and methodology used to produce and validate the test results, and the pricing model used to calculate the price/performance.

CPI server&storage EXASOL	**CPI Phoenix IQ-201 using EXASolution 2.0**	TPC-H Rev. 2.6.2 Report Date **April 2, 2008**

Total System Cost	Composite Query per Hour Metric	Price / Performance
$ 574,565	**547,205.4** QphH@300GB	**$ 1.05** $ / QphH@300GB

Database Size	Database Manager	Operating System	Other Software	Availability Date
300 GB	**EXASolution 2.0**	**EXACluster OS 1.3**	**C++ Compiler**	**April 2, 2008**



Database Load Time = 0.96	Included Backup: N	Data Storage / Database Size = 23.56
RAID (Base tables only): Y	RAID (Base tables and auxiliary data structures): Y	RAID (All): Y

System Configuration:

26 x CPI Phoenix IQ-201 server, each with:
2 Intel XEON X5365 QC 3.0 GHz processors (each is 1 chip, 4 cores, 4 threads)
12 GB RAM
2x 146 GB (10k rpm) internal SATA disks

Total Storage: 7,070 GB
(1 GB = 1024 * 1024 * 1024 bytes)

Description	Part Number	Pricing	Unit Price	Qty	Extended Price	3 yr. Maint. Price
System Hardware						
CPI-Phoenix-Server 2x 3.0 Ghz Quadcore CPUs (Intel® Xeon 5365) 16 FBDIMM-Slots 12x 3.5˝ SAS/S-ATA HotSwap Slots 2x Ethernet Ports GBit 1x IPMI 2.0 Server Management	IQ-201	1	5,602	26+3	162,458	(3 spares)
2GB FBDIMM-667MHz	RAM-02	1	163	156+16	28,036	(16 spares)
3Ware 4 Port SATA Controller (+BBU incl.)	CTR-04	1	594	52+6	34,452	(6 spares)
150 GB S-ATA150 10.000 rpm	HD-150	1	220	52+6	12,760	(6 spares)
Ethernet Cable (RJ45 Cat-5E) 7 feet	CBL-07	1	8	52+6	464	(6 spares)
On-site replenishment within 7 days throughout the 3-year maintenance period	SUP-01	1	Incl.	-	-	-
CPI Server Discount (20%)					-47,634	
HP ProCurve 5406zl + redundant power supply (1000Base-T Ethernet Switch, 72 ports)	HPE-072	1		1	9,967	
HP 3 year 24x7, 4 hours Chassis 6 HW Support	HPE-072-SUP	1				2,437
				Subtotal	**$ 200,503**	**$ 2,437**
Storage						
No external storage required						
Software						
EXASolution 2.0 licence for 3 years	EXA-26N-12G	2		1	395,767	
EXASOL Discount (10%)		2			-39,577	
EXASOL Premium Support (3.9%)	EXA-SUP-P	2				15,435
				Subtotal	**$ 356,190**	**15,435$**
				Total	**$ 556,693,**	**17,872$**
				3-Year Cost of Ownership		**$ 574,565**
				QphH Rating:		**547,205.4**
				$/QphH@300GB:		**1.05**

Price Key: 1-CPI, contact: Peter Markgraf, pmarkgraf@cpigmbh.de
2-EXASOL, contact: Olga Sapozhnykova, sales@exasol.com

All discounts are based on list prices and for similar quantities and configurations.

Results independently audited by: Francois Raab of InfoSizing, Inc. (www.sizing.com)

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 the TPC at pricing@tpc.org.

Numerical Quantities

Measurement Results

Database Scale Factor	300 GB
Total Data Storage / Database Size	23.56
Start of Database Load	12:09:44
End of Database Load	13:07:15
Database Load Time	57m 31s
Query Streams for Throughput Test	11
TPC-H Power	376,927.3
TPC-H Throughput	794,407.3
TPC-H Composite Query-per-Hour Metric (QphH@300GB)	547,205.4
Total System Price Over 3 Years	$574,565
TPC-H Price/ Performance Metric ($/QphH@300GB)	$1.05

Measurement Interval

Measurement Interval in Throughput Test (Ts)	329 seconds

Duration of Stream Execution

Stream ID	Seed	Start Date	Start Time	End Date	End Time	Duration
Stream 0	331130715	2008-03-31	13:07:24	2008-03-31	13:09:02	1 mins:38secs
Stream 1	331130716	2008-03-31	13:09:06	2008-03-31	13:14:09	5 mins:03secs
Stream 2	331130717	2008-03-31	13:09:06	2008-03-31	13:14:02	4 mins:56secs
Stream 3	331130718	2008-03-31	13:09:07	2008-03-31	13:14:17	5 mins:10secs
Stream 4	331130719	2008-03-31	13:09:10	2008-03-31	13:14:17	5 mins:07secs
Stream 5	331130720	2008-03-31	13:09:11	2008-03-31	13:14:23	5 mins:12secs
Stream 6	331130721	2008-03-31	13:09:11	2008-03-31	13:14:30	5 mins:19secs
Stream 7	331130722	2008-03-31	13:09:13	2008-03-31	13:14:31	5 mins:18secs
Stream 8	331130723	2008-03-31	13:09:16	2008-03-31	13:14:30	5 mins:14secs
Stream 9	331130724	2008-03-31	13:09:17	2008-03-31	13:14:28	5 mins:11secs
Stream 10	331130725	2008-03-31	13:09:18	2008-03-31	13:14:31	5 mins:13secs
Stream 11	331130726	2008-03-31	13:09:19	2008-03-31	13:14:35	5 mins:16secs
Refresh		2008-03-31	13:09:03	2008-03-31	13:14:27	5 mins:24secs

TPC-H Timing Intervals (in seconds)

Query	Power	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7	Stream 8	Stream 9	Stream 10	Stream 11	Min S_i	Avg S_i	Max S_i
1	5.0	43.1	42.9	37.0	40.8	40.0	41.9	38.5	38.7	41.5	40.1	43.1	37.0	40.7	43.1
2	0.9	6.8	3.9	6.2	6.7	4.4	5.3	5.6	5.9	4.8	6.3	5.0	3.9	5.5	6.8
3	2.7	8.1	14.6	13.7	11.9	15.3	13.7	12.5	15.3	13.2	11.1	13.7	8.1	13.0	15.3
4	1.1	7.2	7.9	6.6	5.9	6.7	7.7	6.4	6.7	6.9	7.3	6.5	5.9	6.9	7.9
5	4.8	14.3	14.9	13.1	8.4	13.6	18.1	15.0	13.4	13.8	8.2	14.9	8.2	13.4	18.1
6	1.0	7.4	1.8	8.8	7.9	7.2	7.2	7.9	7.8	6.7	7.7	5.3	1.8	6.9	8.8
7	3.7	19.5	16.0	19.1	14.7	18.8	21.1	17.6	19.9	17.4	17.8	9.3	9.3	17.4	21.1
8	4.2	10.0	10.1	4.1	10.8	13.3	10.7	13.5	11.4	8.1	10.4	12.7	4.1	10.5	13.5
9	15.7	46.5	48.5	49.4	53.7	46.3	55.1	52.1	59.1	58.4	60.3	61.8	46.3	53.8	61.8
10	2.8	15.3	13.7	15.4	14.2	18.4	12.6	16.1	16.6	13.4	12.0	14.0	12.0	14.7	18.4
11	4.2	8.5	8.8	8.8	9.1	10.0	9.8	12.5	5.6	11.6	9.7	10.6	5.6	9.5	12.5
12	2.6	15.6	14.3	13.8	12.2	14.9	13.8	8.2	13.6	12.4	11.1	14.6	8.2	13.1	15.6
13	3.1	17.0	15.2	18.2	14.6	15.3	17.6	16.1	18.5	15.5	17.2	16.3	14.6	16.5	18.5
14	3.4	14.0	7.6	11.6	12.8	14.2	12.6	12.9	9.5	9.8	11.6	12.9	7.6	11.8	14.2
15	4.5	17.1	15.4	15.8	18.5	18.2	15.9	16.9	17.6	19.4	17.6	18.6	15.4	17.4	19.4
16	5.9	10.5	11.1	11.5	12.3	12.2	11.2	13.2	12.9	9.1	13.7	11.5	9.1	11.7	13.7
17	1.0	3.4	1.1	4.5	2.8	5.0	3.1	4.5	3.4	3.5	4.2	5.6	1.1	3.7	5.6
18	6.0	16.5	17.2	18.6	18.8	15.3	17.0	16.2	11.5	15.9	14.5	17.6	11.5	16.3	18.8
19	1.5	3.7	4.2	4.6	7.0	4.0	4.1	3.2	2.6	4.4	5.9	2.0	2.0	4.1	7.0
20	2.2	10.9	12.2	11.8	7.0	10.6	7.3	11.1	9.5	10.6	8.8	8.9	7.0	9.9	12.2
21	1.8	4.5	12.7	14.0	14.1	6.7	10.1	14.8	12.1	11.4	14.3	9.1	4.5	11.3	14.8
22	0.4	2.9	2.1	3.8	2.4	2.0	2.9	2.9	2.6	3.2	3.1	2.1	2.0	2.7	3.8
RF1	16.8	37.4	23.5	19.8	23.6	23.9	21.2	21.6	23.1	24.3	22.2	22.5	19.8	23.9	37.4
RF2	2.7	5.4	5.8	4.8	6.0	5.4	4.5	6.1	6.2	7.6	5.2	3.7	3.7	5.5	7.6

Table of Contents

ABSTRACT **3**

Overview 3
Standard and Executive Summary Statements 3
Auditor 3

TABLE OF CONTENTS **8**

1 GENERAL ITEMS **12**

1.1 Test Sponsor 12
1.2 Parameter Settings 12
1.3 Configuration Items 12

2 CLAUSE 1: LOGICAL DATABASE DESIGN RELATED ITEMS **14**

2.1 Database Definition Statements 14
2.2 Physical Organization 14
2.3 Horizontal Partitioning 14
2.4 Replication 14

3 CLAUSE 2: QUERIES AND REFRESH FUNCTIONS **15**

3.1 Query Language 15
3.2 Verifying Method for Random Number Generation 15
3.3 Generating Values for Substitution Parameters 15
3.4 Query Text and Output Data from Qualification Database 15
3.5 Query Substitution Parameters and Seeds Used 15
3.6 Isolation Level 15
3.7 Source Code of Refresh Functions 15

4 CLAUSE 3: DATABASE SYSTEM PROPERTIES **16**

4.1 ACID Properties 16
4.2 Atomicity Requirements 16
4.3 Consistency Requirements 16
4.4 Isolation Requirements 17
4.5 Durability Requirements 19

5 CLAUSE 4: SCALING AND DATABASE POPULATION **20**

5.1 Ending Cardinality of Tables 20
5.2 Distribution of Tables and Logs Across Media 20
5.3 Mapping of Database Partitions/Replication 20
5.4 Implementation of RAID 20
5.5 DBGEN Modifications 21
5.6 Data Base Load Time 21
5.7 Data Storage Ratio 21
5.8 Database Load Mechanism Details and Illustration 21
5.9 Qualification Database Configuration 22
5.10 Base Data Comparison 22
5.11 Referential Integrity of Initial Database Load 22

6 CLAUSE 5: PERFORMANCE METRICS AND EXECUTION RULES RELATED ITEMS **23**

6.1 System Activity between Load and Performance Tests 23
6.2 Steps in the Power Test 23
6.3 Timing Interval for Each Query and Refresh Functions 23
6.4 Number of Streams for the Throughput Test 23
6.5 Start and End Date/Time of Each Query Stream 23
6.6 Total Elapsed Time of the Measurement Interval 23
6.7 Refresh Function Start Date/Time and Finish Date/Time 23
6.8 Timing Intervals for Each Query and Each Refresh Function for Each Stream 23
6.9 Performance Metrics 24

6.10 The Performance Metric and Numerical Quantities from Both Runs 24
6.11 System Activity between Performance Tests 24
7 CLAUSE 6: SUT AND DRIVER IMPLEMENTATION RELATED ITEMS 25
7.1 Driver 25
7.2 Implementation Specific Layer (ISL) 25
7.3 Profile-Directed Optimization 25
8 CLAUSE 7: PRICING 26
8.1 Hardware and Software Used in the Priced System 26
8.2 Total Three Year Price 26
8.3 Availability Date 26
9 CLAUSE 8: AUDITOR'S INFORMATION AND ATTESTATION LETTER 27
9.1 Auditor's Report 27
APPENDIX A: SYSTEM AND DATABASE CONFIGURATIONS AND PARAMETERS 28
APPENDIX B: DATABASE BUILD SCRIPTS 29
APPENDIX C: ACID SCRIPTS 31
APPENDIX D: QUERY TEXT AND QUERY OUTPUT 69
APPENDIX E: SEED AND INPUT PARAMETERS 80
APPENDIX F: BENCHMARK PROGRAMS AND SCRIPTS 82
APPENDIX G: PRICING INFORMATION 100

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Benchmark Sponsor:

Peter Markgraf
General Manager Sales & Marketing
Computer Partner Handels GmbH
Kapellenstr. 11
D-85622 Feldkirchen

Mathias Golombek
Primary TPC Representative
EXASOL AG
Neumeyerstr. 48
D-90411 Nürnberg

March 31, 2008

I verified the TPC Benchmark™ H performance of the following configuration:

Platform: **CPI Phoenix IQ-201, 26-node Cluster**

Database Manager: **EXASolution 2.0**

Operating System: **EXACluster OS 1.3**

The results were:

CPU (Speed)	Memory	Disks	**QphH@300GB**
CPI Phoenix IQ-201, 12-node cluster (each node with)			
2 x Intel XEON X5365 Quad (3.0GHz)	12GB Main	2 x 146GB SATA	**547,205.4**

In my opinion, this performance result was produced in compliance with the TPC's requirements for the benchmark. The following verification items were given special attention:

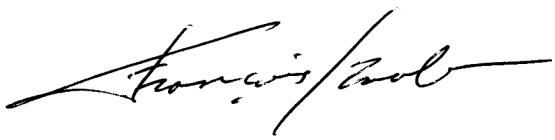
- The database records were defined with the proper layout and size
- The database population was generated using DBGEN
- The database was properly scaled to 300GB and populated accordingly
- The compliance of the database auxiliary data structures was verified

- The database load time was correctly measured and reported
- The required ACID properties were verified and met
- The query input variables were generated by QGEN
- The query text was produced using minor modifications
- The execution of the queries against the SF1 database produced compliant answers
- A compliant implementation specific layer was used to drive the tests
- The throughput tests involved 11 query streams
- The ratio between the longest and the shortest query was such that no query timing was adjusted
- The execution times for queries and refresh functions were correctly measured and reported
- The repeatability of the measured results was verified
- The required amount of database log was configured
- The system pricing was verified for major components and maintenance
- The major pages from the FDR were verified for accuracy

Additional Audit Notes:

None.

Respectfully Yours,



François Raab
President

1 General Items

1.1 Test Sponsor

A statement identifying the benchmark sponsor(s) and other participating companies must be provided.

This TPC-H benchmark is sponsored by CPI Computer Partner Handels GmbH and EXASOL AG. The benchmark implementation was developed and engineered by EXASOL AG.

1.2 Parameter Settings

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

- *Database Tuning Options*
- *Optimizer/Query execution options*
- *Query processing tool/language configuration parameters*
- *Recovery/commit options*
- *Consistency/locking options*
- *Operating system and configuration parameters*
- *Configuration parameters and options for any other software component incorporated into the pricing structure*
- *Compiler optimization options*

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

Details of system and database configuration and parameters are provided in Appendix A.

1.3 Configuration Items

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

- *Number and type of processors.*
- *Size of allocated memory, and any specific mapping/partitioning of memory unique to the test.*
- *Number and type of disk units (and controllers, if applicable).*
- *Number of channels or bus connections to disk units, including their protocol type.*
- *Number of LAN (e.g. Ethernet) Connections, including routers, workstations, terminals, etc., that were physically used in the test or are incorporated into the pricing structure.*
- *Type and the run-time execution location of software components (e.g., DBMS, query processing tools/languages, middle-ware components, software drivers, etc.).*

The System Under Test (SUT), depicted in Figure 1.1, that was used to obtain the results in this benchmark consists of a cluster of CPI Phoenix IQ-201 servers, each with:

	System components
2	QC CPU (Intel® Xeon X5365, 3GHz)
12	GB RAM
2	146 GB SATA 10.000 rpm
2	3Ware SATA controller (BBU included)
2	Ethernet Ports
1	HP ProCurve 5406zl (1000Base-T Ethernet Switch, 72 ports)

Network is Gigabit Ethernet, there is one physical network for initial data distribution, synchronization and administration. Each server contains 2 physical disks. 1 pair of 2 disks is mirrored among the controllers (Software RAID 1).

Priced configuration and benchmarked configuration are identical.

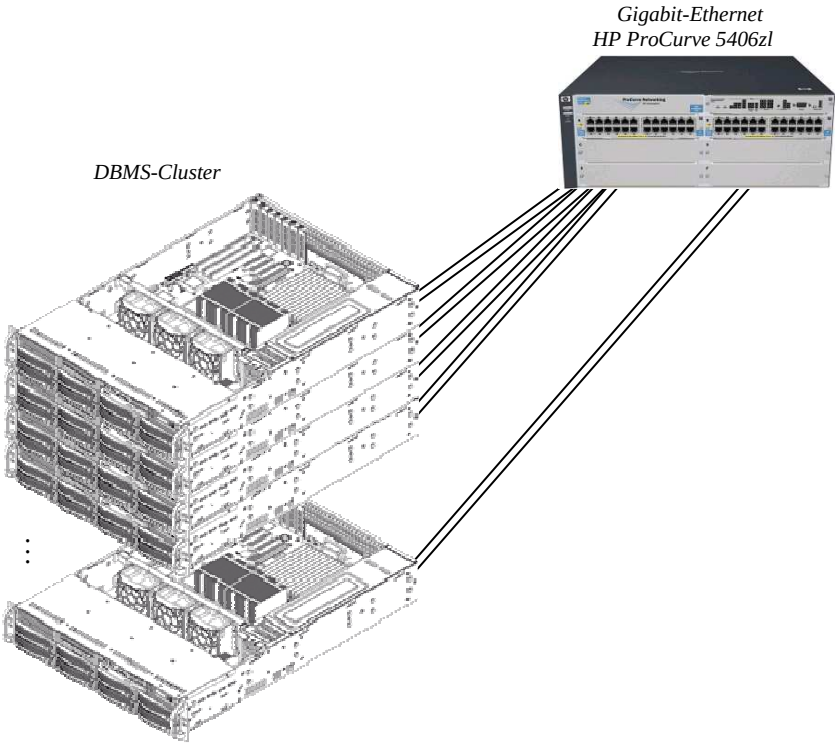


Figure 1.1: Benchmarked and priced system configuration

2 Clause 1: Logical Database Design Related Items

2.1 Database Definition Statements

Listings must be provided for all table definition statements and all other statements used to set up the test and qualification databases (8.1.2.1).

Appendix B contains the build scripts that define the tables and indices for the TPC-H database.

2.2 Physical Organization

The physical organization of tables and indices, within the test and qualification databases, must be disclosed. If the column ordering of any table is different from that specified in Clause 1.4, it must be noted.

Physical organization requires no user input. All the database data is placed on the same partition.

2.3 Horizontal Partitioning

Horizontal partitioning of tables and rows in the test and qualification databases (see Clause 1.5.4) must be disclosed.

Horizontal partitioning is used. The data is automatically distributed on the cluster nodes using a hash algorithm. The columns used for the hashing are controlled by DDL statements (see Appendix B).

2.4 Replication

Any replication of physical objects must be disclosed and must conform to the requirements of Clause 1.5.6.

No objects were physically replicated.

3 Clause 2: Queries and Refresh Functions

3.1 Query Language

The query language used to implement the queries must be identified.

SQL was the query language uniquely used throughout this benchmark.

3.2 Verifying Method for Random Number Generation

The method of verification for the random number generation must be described unless the supplied DBGEN and QGEN were used.

TPC supplied versions 2.6.2 of DBGEN and version 2.6.2 of QGEN were used in this benchmark. The DBGEN tool was patched by EXASOL changing the output format (see section 5.5).

3.3 Generating Values for Substitution Parameters

The method used to generate values for substitution parameters must be disclosed. If QGEN is not used for this purpose, then the source code of any non-commercial tool used must be disclosed. If QGEN is used, the version number, release number, modification number, and patch level of QGEN must be disclosed.

QGEN version 2.6.2 was used to generate the substitution parameters.

3.4 Query Text and Output Data from Qualification Database

The executable query text used for query validation must be disclosed along with the corresponding output data generated during the execution of the query text against the qualification database. If minor modifications (see Clause 2.2.3) have been applied to any functional query definition or approved variants in order to obtain executable query text, these modifications must be disclosed and justified. The justification for a particular minor query modification can apply collectively to all queries for which it has been used. The output data for the power and throughput tests must be made available electronically upon request.

Appendix D contains the actual query text and query output. Minor modifications have been applied and were commented as such.

3.5 Query Substitution Parameters and Seeds Used

The query substitution parameters used for all performance tests must be disclosed in tabular format, along with the seeds used to generate these parameters.

Appendix E contains the seed and query substitution parameters.

3.6 Isolation Level

The isolation level used to run the queries must be disclosed. If the isolation level does not map closely to the levels defined in Clause 3.4, additional descriptive detail must be provided.

The queries and transactions were run with the isolation level 3 (phenomena P1 through P4, as defined in the TPC-H specification, are prevented from occuring).

3.7 Source Code of Refresh Functions

The details of how the refresh functions were implemented must be disclosed (including source code of any non-commercial program used).

Appendix F contains the source code of the refresh functions.

4 Clause 3: Database System Properties

4.1 ACID Properties

The ACID (Atomicity, Consistency, Isolation, and Durability) properties of transaction processing systems must be supported by the system under test during the timed portion of this benchmark. Since TPC-H is not a transaction processing benchmark, the ACID properties must be evaluated outside the timed portion of the test.

Appendix C contains the source code of the ACID test scripts.

4.2 Atomicity Requirements

The system under test must guarantee that transactions are atomic; the system will either perform all individual operations on the data, or will assure that no partially completed operations leave any effects on the data.

4.2.1 Atomicity of the Completed Transactions

Perform the ACID Transaction for a randomly selected set of input data and verify that the appropriate rows have been changed in the ORDERS, LINEITEM, and HISTORY tables.

The following steps were performed to verify the atomicity of the completed ACID transactions:
1. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for a randomly selected order key.
2. One ACID Transaction was performed using the order key from step 1.
3. The ACID Transaction was committed.
4. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for the same order key.
5. It was verified that the appropriate rows had been changed.

4.2.2 Atomicity of Aborted Transactions

Perform the ACID Transaction for a randomly selected set of input data, substituting a ROLLBACK of the transaction for the COMMIT of the transaction. Verify that the appropriate rows have not been changed in the ORDERS, LINEITEM, and HISTORY tables.

The following steps were performed to verify the atomicity of the completed ACID transactions:
1. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for a randomly selected order key.
2. One ACID Transaction was performed using the order key from step 1. The transaction was stopped prior to the commit.
3. The ACID Transaction was rolled back.
4. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for the same order key.
5. It was verified that the appropriate rows had not been changed.

4.3 Consistency Requirements

Consistency is the property of the application that requires any execution of transactions to take the database from one consistent state to another. A consistent state for the TPC-H database is defined to exist when:

```
      O_TOTALPRICE =
            SUM(trunc(trunc(L_EXTENDEDPRICE*(1-L_DISCOUNT),2)*(1+L_TAX),2))
For each ORDER and LINEITEM defined by (O_ORDERKEY = L_ORDERKEY).
```

4.3.1 Consistency Test

Verify that ORDERS and LINEITEM tables are initially consistent, submit the prescribed number of ACID Transactions with randomly selected input parameters, and re-verify the consistency of the ORDERS and LINEITEM.

The following queries were executed before and after the durability tests to show that the database was always in a consistent state both initially and after submitting transactions:

```
SELECT *
FROM (
       SELECT o_orderkey, o_totalprice – sum(trunc(trunc(l_extendedprice *
                          (1-l_discount),2)*(1+l_tax),2)) part_res
       FROM orders, lineitem
       WHERE o_orderkey=l_orderkey
       GROUP BY o_orderkey, o_totalprice
) WHERE not part_res=0;
```

EXASolution 2.0 has serializable isolation level with table level lock concurrency control. The ACID Transaction of stream0 was expanded with 5 seconds delay after the update and before commit. Since only one update transaction can execute at any one time, the delay should guaranty that the active update transaction is "in-flight" at the time of the failure. To improve the throughput of the concurrent streams a random delay was used after COMMIT (0-5 seconds) and after transaction conflicts (0-2 seconds).

The following steps were performed to verify the consistency of ACID transactions:
1. The consistency of the ORDERS and LINEITEM tables was verified.
2. 1000 transactions for each of the 11+1 execution streams were prepared.
3. After that at least 100 ACID transactions were submitted from each of 11+1 execution streams.
4. A durability failure was induced by pulling a disk drive or switching off the power to one or more servers.
5. The consistency of the ORDERS and LINEITEM tables was re-verified.

4.4 Isolation Requirements

Operations of concurrent transactions must yield results, which are indistinguishable from the results, which would be obtained by forcing each transaction to be serially executed to completion in some order.

The steps of the isolation tests were adopted to the EXASOL isolation environment.

4.4.1 Isolation Test 1 – Read-Write Conflict with Commit

Demonstrate isolation for the read-write conflict of a read-write transaction and a read-only transaction when the read-write transaction is committed

The following steps were performed to satisfy the test of isolation for a read-only and a read-write committed transaction:

1. Start a query and verify that the row was retrieved.
2. Start an update transaction, read and update the same row. Wait before commit.
3. Start the same query and verify that the row retrieved has not changed.
4. Commit the update transaction
5. Start the same query and verify that the new row is retrieved

4.4.2 Isolation Test 2 – Read-Write Conflict with Rollback

Demonstrate isolation for the read-write conflict of a read-write transaction and a read-only transaction when the read-write transaction is rolled back.

The following steps were performed to satisfy the test of isolation for a read-only and a rolled back read-write transaction:

1. Start a query and verify that the row was retrieved.
2. Start an update transaction, read and update the same row. Wait before commit.
3. Start the same query and verify that the row retrieved has not changed.
4. Rollback the update transaction
5. Start the same query and verify that the old row (step 1) is retrieved

4.4.3 Isolation Test 3 – Write-Write Conflict with Commit

Demonstrate isolation for the write-write conflict of two update transactions when the first transaction is committed.

The following steps were performed to verify isolation of two update transactions:

1. Start a query and verify that the row was retrieved.
2. Start an update transaction, read and update the same row. Wait before commit.

3. Start another update transaction, read and try to update the same row and verify that the transaction is forced to rollback.
4. commit the update transaction
5. Start the same query and verify that the new row is retrieved

4.4.4 Isolation Test 4 – Write-Write Conflict with Rollback

Demonstrate isolation for the write-write conflict of two update transactions when the first transaction is rolled back.

The following steps were performed to verify isolation of two update transactions after the first one is rolled back:

1. Start a query and verify that the row was retrieved.
2. Start an update transaction, read and update the same row. Wait before commit.
3. Start another update transaction, read and try to update the same row and verify that the transaction is forced to rollback.
4. Rollback the update transaction
5. Start the same query and verify that the new row retrieved in step 1 has not changed

4.4.5 Isolation Test 5 – Concurrent Read and Write Transactions on Different Tables

Demonstrate the ability of read and write transactions affecting different database tables to make progress concurrently.

The following steps were performed to demonstrate the ability of read and write transactions affecting different tables to make progress concurrently:

1. Create a copy of tables Lineitem and Orders. The original tables are referred to as TablePair A. The duplicate tables are referred to as TablePair B.
2. Start a query against TablePair A and verify that the row was retrieved.
3. Start a query against TablePair B and verify that the row was retrieved.
4. Start an update transaction, read and update the same row in TablePair A. Wait before commit.
5. Start an update transaction, read and update the same row in TablePair B and commit.
6. Commit the update transaction against TablePair A.
7. Start a query against TablePair A and verify that the updated row was retrieved.
8. Start a query against TablePair B and verify that the updated row was retrieved.

4.4.6 Isolation Test 6 – Update Transactions during Continuous Read-Only Query Stream

Demonstrate the continuous submission of arbitrary (read-only) queries against one or more tables of the database does not indefinitely delay update transactions affecting those tables from making progress.

The following query was used:

```
SELECT  l1.l_quantity,
        SUM(l2.l_extendedprice),
        SUM(l3.l_extendedprice),
        SUM(l3.l_quantity)
FROM    lineitem l1, lineitem l2, lineitem l3
WHERE   l1.l_shipdate <= DATE '1998-12-01' -0
        AND l1.l_orderkey = l2.l_orderkey
        AND l1.l_linenumber = l2.l_linenumber
        AND l1.l_extendedprice = l3.l_extendedprice
        AND l3.l_quantity < 30
GROUP BY l1.l_quantity;
COMMIT;
```

1. A Transaction, T1, which executed the above query against the qualification database, was started using a randomly selected DELTA.
2. An ACID Transaction, T2, was started for a randomly selected O_KEY, L_KEY and DELTA.
3. T2 completed and appropriate rows in the ORDERS, LINEITEM and HISTORY tables had been changed.
4. T1 was still executing.
5. Transaction T1 completed executing the query.

4.5 Durability Requirements

The tested system must guarantee durability: the ability to preserve the effects of committed transactions and insure database consistency after recovery from any one of the failures listed in Clause 3.5.2.

4.5.1 Permanent Unrecoverable Failure of Any Durable Medium

Guarantee the database and committed updates are preserved across a permanent irrecoverable failure of any single durable medium containing TPC-H database tables or recovery log tables.

During the durability test, one disk was removed. The test continued uninterrupted, because of the RAID protection (see also section 1.3). Upon replacement/re-insert of the drive, the data files were rebuild to a consistent state by the RAID controller. The durability success file and the HISTORY table were compared and the counts matched.

4.5.2 System Crash

Guarantee the database and committed updates are preserved across an instantaneous interruption (system crash/system hang) in processing which requires the system to reboot to recover.

The system crash and memory failure tests were combined. Two system crashs were performed by turning off the power during the durability test: one test for only one node and the other test for the whole cluster.

When power was restored, in both cases the system rebooted automatically and the database was restarted manually. The durability success file and the HISTORY table were compared and the counts matched.

4.5.3 Memory Failure

Guarantee the database and committed updates are preserved across failure of all or part of memory (loss of contents). See the previous section.

The system crash and memory failure tests were combined as explained in section 4.5.2.

5 Clause 4: Scaling and Database Population

5.1 Ending Cardinality of Tables

The cardinality (e.g., the number of rows) of each table of the test database, as it existed at the completion of the database load (see clause 4.2.5) must be disclosed.

Table 5.1 lists the TPC Benchmark H defined tables and the row count for each table as they existed upon completion of the build.

Table Name	Cardinality
LINEITEM	1,799,989,091
ORDER	450,000,000
PARTSUPP	240,000,000
PART	60,000,000
CUSTOMER	45,000,000
SUPPLIER	3,000,000
NATION	25
REGION	5

Table 5.1: Initial Number of Rows

5.2 Distribution of Tables and Logs Across Media

The distribution of tables and logs across all media must be explicitly described for the tested and priced systems.

Each server contains 2 physical disks. 1 pair of 2 disks is mirrored among the controllers (RAID 1). Controller cache is switched on.

The resulting device is divided into 4 partitions (see Table 5.2). All benchmark- and database-relevant data is stored in partition `Data` (Tables, Indexes, Temp, Logs, Flat Files).

Partition	Size
Boot	0.1 GB
System	50 GB
Swap	26 GB
Data	70 GB

Table 5.2: Disk layout schema for one DBMS Cluster node

5.3 Mapping of Database Partitions/Replication

The mapping of database partitions/replications must be explicitly described.

No database partitions/replications were used in this benchmark.

5.4 Implementation of RAID

Implementations may use some form of RAID to ensure high availability. If used for data, auxiliary storage (e.g. indexes) or temporary space, the level of RAID must be disclosed for each device.

Please refer to chapter 5.2.

5.5 DBGEN Modifications

The version number, release number, modification number, and patch level of DBGEN must be disclosed. Any modifications to the DBGEN (see Clause 4.2.1) source code must be disclosed. In the event that a program other than DBGEN was used to populate the database, it must be disclosed in its entirety.

TPC supplied versions 2.6.2 of DBGEN with a minor EXASOL patch was used in this benchmark. The patch resulted in the following changes of the output format:

- Quotes for strings
- Separators

See Appendix F for details.

5.6 Data Base Load Time

The database load time for the test database (see Clause 4.3) must be disclosed.

The database load time was 1h55m19sec.

5.7 Data Storage Ratio

The data storage ratio must be disclosed. It is computed by dividing the total data storage of the priced configuration (expressed in GB) by the size chosen for the test database as defined in 4.1.3.1. The ratio must be reported to the nearest 1/100th, rounded up.

Disk Type	GB per disk*	# of disks	Total (GB)**
Internal	146 GB	52	7070.60

Scale Factor	Data Storage Ratio
300	23.56

** Disk manufacturer definition of 1 GB is 10^9 bytes*
*** In this calculation 1 GB is defined as 2^{30} bytes*

5.8 Database Load Mechanism Details and Illustration

The details of the database load must be disclosed, including a block diagram illustrating the overall process. Disclosure of the load procedure includes all steps, scripts, input and configuration files required to completely reproduce the test and qualification databases.

The database was loaded using data generation stored on the flat files all on the tested and priced configuration. DBGEN was used to create the flat files.

The block diagram in figure 5.8.1 describes the process used to load the database.

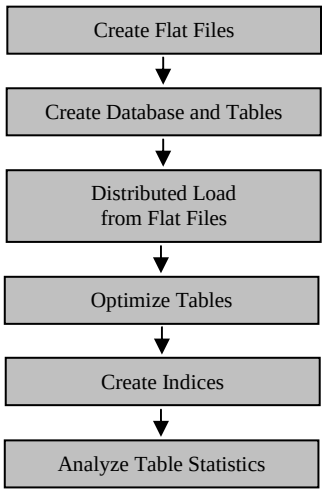


Figure 5.8.1: Database Load Process

5.9 Qualification Database Configuration

Any differences between the configuration of the qualification database and the test database must be disclosed.

The qualification database used identical scripts to create and load the data with changes to adjust for the database scale factor.

5.10 Base Data Comparison

Verify that the rows in the loaded database after the performance test are correct by comparing some small number of rows extracted at random from any two files of the corresponding Base, Insert and Delete reference data set files for each table and the corresponding rows of the database.

Verified according to the specification.

5.11 Referential Integrity of Initial Database Load

Verify referential integrity in the database after the initial load.

Verified according to the specification.

6 Clause 5: Performance Metrics and Execution Rules Related Items

6.1 System Activity between Load and Performance Tests

Any system activity on the SUT that takes place between the conclusion of the load test and the beginning of the performance test must be fully disclosed.

There is no activity on the SUT between the conclusion of the load test and the beginning of the performance test. Benchmark scripts are started just after the load was completed.

6.2 Steps in the Power Test

The details of the steps followed to implement the power test (e.g., system boot, database restart, etc.) must be disclosed.

The following steps were used to implement the power test:

1. Database was started
2. RF1 Refresh Transaction
3. Stream 00 Execution
4. RF2 Refresh Transaction

6.3 Timing Interval for Each Query and Refresh Functions

The timing intervals (see Clause 5.3.6) for each query of the measured set for both refresh functions must be reported for the power test.

The timing intervals for each query and both update functions are given in the Numerical Quantities Summary earlier in this document.

6.4 Number of Streams for the Throughput Test

The number of execution streams used for the throughput test must be disclosed.

7 query streams and 1 refresh stream were used for the throughput test.

6.5 Start and End Date/Time of Each Query Stream

The start time and finish time for each query stream must be reported for the throughput test.

The throughput test start time and finish time for each stream are given in the Numerical Quantities Summary earlier in this document.

6.6 Total Elapsed Time of the Measurement Interval

The total elapsed time of the measurement interval (see Clause 5.3.5) must be reported for the throughput test.

The total elapsed time of the throughput test is given in the Numerical Quantities Summary earlier in this document.

6.7 Refresh Function Start Date/Time and Finish Date/Time

Start and finish time for each update function in the update stream must be reported for the throughput test.

The refresh function start and finish date/time are given in the Numerical Quantities Summary earlier in this document.

6.8 Timing Intervals for Each Query and Each Refresh Function for Each Stream

The timing intervals (see Clause 5.3.6) for each query of each stream and for each refresh function must be reported for the throughput test.

The timing intervals for each query and each update function are given in the Numerical Quantities Summary earlier in this document.

6.9 Performance Metrics

The computed performance metric, related numerical quantities and price performance metric must be reported.

The performance metrics, and the numbers, on which they are based, is given in the Numerical Quantities Summary earlier in this document.

6.10 The Performance Metric and Numerical Quantities from Both Runs

A description of the method used to determine the reproducibility of the measurement results must be reported. This must include the performance metrics (QppH and QthH) from reproducibility runs.

Run ID	QppH@300GB	QptH@300GB	QphH@300GB
Run 1	376,927.3	794,407.8	547,205.4
Run 2	379,031.2	792,000.0	547,898.4
% Difference	0.56 %	0.56 %	0.13 %

6.11 System Activity between Performance Tests

Any activity on the SUT that takes place between the conclusion of Run 1 and the beginning of Run 2 must be disclosed.

There was no system activity between Run 1 and Run 2.

7 Clause 6: SUT and Driver Implementation Related Items

7.1 Driver

A detailed description of how the driver performs its functions must be supplied, including any related source code or scripts. This description should allow an independent reconstruction of the driver.

All stream executions are performed by a script. QGEN is used to produce query text.

For each power-test run:

1. The shell script for RF1 is started
2. Then the queries as generated by QGEN are submitted in the order defined by Clause 5.3.5.4 from the driver.
3. The shell script for RF2 is started.

For each throughput-test run:

1. The queries as generated by QGEN are submitted in the order defined by Clause 5.3.5.4 from the driver in several streams.
2. Parallely, pairs of RF1/RF2 are executed sequentially in one update stream.

The source code of the used scripts are disclosed on Appendix F.

7.2 Implementation Specific Layer (ISL)

If an implementation specific layer is used, then a detailed description of how it performs its functions must be supplied, including any related source code or scripts. This description should allow an independent reconstruction of the implementation-specific layer.

This benchmark test did not make use of an ISL.

7.3 Profile-Directed Optimization

If profile-directed optimization as described in Clause 5.2.9 is used, such use must be disclosed.

Profile-directed optimization was not used in this benchmark test.

8 Clause 7: Pricing

8.1 Hardware and Software Used in the Priced System

A detailed list of hardware and software used in the priced system must be reported. Each item must have vendor part number, description, and release/revision level, and either general availability status or committed delivery date. If package-pricing is used, contents of the package must be disclosed. Pricing source(s) and effective date(s) of price(s) must also be reported.

A detailed list of hardware and software used in the priced system is included in the pricing sheet in the executive summary. All prices are currently effective. Third-party price quotations are included in Appendix G.

8.2 Total Three Year Price

The total 3-year price of the entire configuration must be reported including: hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.

A detailed pricing sheet of all the hardware and software used in this configuration and the 3-year maintenance costs, demonstrating the computation of the total 3-year price of the configuration, is included in the executive summary at the beginning of this document.

8.3 Availability Date

The committed delivery date for general availability of products used in the priced calculations must be reported. When the priced system includes products with different availability dates, the availability date reported on the executive summary must be the date by which all components are committed to being available. The full disclosure report must report availability dates individually for at least each of the categories for which a pricing subtotal must be provided.

Component	Availability Date
Cluster Hardware	Now (date of publication)
EXASolution 2.0	Now (date of publication)

9 Clause 8: Auditor's Information and Attestation Letter

9.1 Auditor's Report

The auditor's agency name, address, phone number, and Attestation letter with a brief audit summary report indicating compliance must be included in the full disclosure report. A statement should be included specifying who to contact in order to obtain further information regarding the audit process.

This implementation of the TPC Benchmark H was audited by Francois Raab for InfoSizing.

Further information regarding the audit process may be obtained from:

Francois Raab
InfoSizing, Inc.
125 West Monroe Street
Colorado Springs, CO 80907
(719) 473-7555
(719) 473-7554
Email: francois@sizing.com

The auditor's attestation letter is included after the table of contents.

TPC Benchmark H Full Disclosure Report and other information can be downloaded from the Transaction Processing Performance Council website at www.tpc.org.

Appendix A: System and Database Configurations and Parameters

Kernel-Parameters:

```
kernel.shmmax = 2147483648
kernel.shmall = 8388608
kernel.shmmni = 8388608
kernel.msgmnb = 1073741824
kernel.msgmni = 256
kernel.msgmax = 1073741824
vm.swappiness = 0
vm.vfs_cache_pressure = 100
vm.overcommit_ratio = 50
vm.overcommit_memory = 0
vm.nr_hugepages = 3000
```

Database Parameters:

```
-dictionarythreshold 450
-replicationborder 3030000
-slb 6000
```

Appendix B: Database Build Scripts

```
# ------------------------------------------------------------------------------------------
# create_schema.sql
# ------------------------------------------------------------------------------------------

CREATE SCHEMA tpc;
OPEN SCHEMA tpc;
CREATE OR REPLACE TABLE NATION  ( N_NATIONKEY DEC(11), N_NAME CHAR(25), N_REGIONKEY DEC(11), N_COMMENT VARCHAR(152) );
CREATE OR REPLACE TABLE REGION  ( R_REGIONKEY DEC(11), R_NAME CHAR(25), R_COMMENT VARCHAR(152) );
CREATE OR REPLACE TABLE PART  ( P_PARTKEY DEC(11), P_NAME VARCHAR(55), P_MFGR CHAR(25), P_BRAND CHAR(10), P_TYPE VARCHAR(25),
             P_SIZE DEC(10), P_CONTAINER CHAR(10), P_RETAILPRICE DECIMAL(12,2), P_COMMENT VARCHAR(23) );
CREATE OR REPLACE TABLE SUPPLIER ( S_SUPPKEY DEC(11), S_NAME CHAR(25), S_ADDRESS VARCHAR(40), S_NATIONKEY DEC(11), S_PHONE
             CHAR(15), S_ACCTBAL DECIMAL(12,2), S_COMMENT VARCHAR(101) );
CREATE OR REPLACE TABLE PARTSUPP ( PS_PARTKEY DEC(11), PS_SUPPKEY DEC(11), PS_AVAILQTY DEC(10), PS_SUPPLYCOST DECIMAL(12,2),
             PS_COMMENT VARCHAR(199) );
CREATE OR REPLACE TABLE CUSTOMER ( C_CUSTKEY DEC(11), C_NAME VARCHAR(25), C_ADDRESS VARCHAR(40), C_NATIONKEY DEC(11), C_PHONE
             CHAR(15), C_ACCTBAL DECIMAL(12,2), C_MKTSEGMENT CHAR(10), C_COMMENT VARCHAR(117) );
CREATE OR REPLACE TABLE ORDERS  ( O_ORDERKEY DEC(11), O_CUSTKEY DEC(11), O_ORDERSTATUS CHAR(1), O_TOTALPRICE DECIMAL(12,2),
             O_ORDERDATE DATE, O_ORDERPRIORITY CHAR(15), O_CLERK CHAR(15), O_SHIPPRIORITY DEC(10), O_COMMENT VARCHAR(79) );
CREATE OR REPLACE TABLE LINEITEM ( L_ORDERKEY DEC(11), L_PARTKEY DEC(11), L_SUPPKEY DEC(11), L_LINENUMBER DEC(10), L_QUANTITY
             DECIMAL(12,2), L_EXTENDEDPRICE DECIMAL(12,2), L_DISCOUNT DECIMAL(12,2), L_TAX DECIMAL(12,2), L_RETURNFLAG
             CHAR(1), L_LINESTATUS CHAR(1), L_SHIPDATE DATE, L_COMMITDATE DATE, L_RECEIPTDATE DATE, L_SHIPINSTRUCT CHAR(25),
             L_SHIPMODE CHAR(10), L_COMMENT VARCHAR(44) );
COMMIT;
ALTER TABLE CUSTOMER set primary join attribute C_CUSTKEY;
ALTER TABLE ORDERS set primary join attribute O_CUSTKEY;
ALTER TABLE LINEITEM set primary join attribute L_ORDERKEY;
ALTER TABLE PARTSUPP set primary join attribute PS_PARTKEY, PS_SUPPKEY;
ALTER TABLE PART set primary join attribute P_PARTKEY;
ALTER TABLE SUPPLIER set primary join attribute S_SUPPKEY;
COMMIT;


# ------------------------------------------------------------------------------------------
# create_user.sql
# ------------------------------------------------------------------------------------------

create user tpcuser identified by tpcuser;
grant create session to tpcuser;
grant create table to tpcuser;
grant create view to tpcuser;
grant create schema to tpcuser;

-- to get debug output of system tables
grant select any dictionary to tpcuser;
commit;


# ------------------------------------------------------------------------------------------
# create_indices.sql
# ------------------------------------------------------------------------------------------
set autocommit off;
open schema tpc;
optimize table nation;
commit;
optimize table region;
commit;
optimize table part;
commit;
optimize table partsupp;
commit;
optimize table supplier;
commit;
optimize table customer;
commit;
optimize table orders;
commit;
optimize table lineitem;
commit;
enforce index lineitem_on_suppkey on lineitem (l_suppkey);
enforce index lineitem_on_partkey_suppkey on lineitem (l_partkey, l_suppkey);
enforce index lineitem_on_partkey on lineitem (l_partkey);
enforce index lineitem_on_orderkey on lineitem (l_orderkey);
enforce index nation_on_nationkey on nation (n_nationkey);
enforce index region_on_regionkey on region (r_regionkey);
enforce index supplier_on_suppkey on supplier (s_suppkey);
enforce index supplier_on_nationkey on supplier (s_nationkey);
enforce index customer_on_custkey on customer (c_custkey);
enforce index customer_on_nationkey on customer (c_nationkey);
enforce index part_on_partkey on part (p_partkey);
enforce index partsupp_on_partkey_suppkey on partsupp (ps_partkey, ps_suppkey);
enforce index partsupp_on_partkey on partsupp (ps_partkey);
enforce index partsupp_on_suppkey on partsupp (ps_suppkey);
enforce local index orders_on_orderkey_l on orders (o_orderkey);
enforce global index orders_on_orderkey_g on orders (o_orderkey);
enforce index orders_on_custkey on orders (o_custkey);
commit;
analyze table part estimate statistics;
analyze table partsupp estimate statistics;
analyze table supplier estimate statistics;
analyze table customer estimate statistics;
analyze table orders estimate statistics;
analyze table lineitem estimate statistics;
analyze table nation estimate statistics;
analyze table region estimate statistics;
commit;
```

```
# ------------------------------------------------------------------------------------------
# load_init_parallel.sh
# ------------------------------------------------------------------------------------------

#!/bin/bash

if [ $# -ne 8 ]; then
    echo "Syntax: $0 <dw> <user> <password> <schema> <SF> <instances-per-node> <dwconn> <dwnodes>"
    exit 1
fi

host=$1
port=$2
user=$3
pass=$4
schema=$5
SF=$6
INST_PN=$7
Loader=$LOADERBINDIR/EXAloader_SYNC
ParLoader=$LOADERBINDIR/EXAloader_PARSYNC

export TPC_SINGLENODE_CSVDIR="/d02_data/csv/single"

TPC_LOCALDIR="/d02_data/csv/local/TPC_Database_SF${SF}_${INST_PN}x_${NODES}N"

NATION_COLUMNS="N_NATIONKEY:DECIMAL(11)|N_NAME:CHAR(25)|N_REGIONKEY:DECIMAL(11)|N_COMMENT:VARCHAR(152)"
REGION_COLUMNS="R_REGIONKEY:DECIMAL(11)|R_NAME:CHAR(25)|R_COMMENT:VARCHAR(152)"
PART_COLUMNS="P_PARTKEY:DECIMAL(11)|P_NAME:VARCHAR(55)|P_MFGR:CHAR(25)|P_BRAND:CHAR(10)|P_TYPE:VARCHAR(25)|P_SIZE:DECIMAL(10)
|P_CONTAINER:CHAR(10)|P_RETAILPRICE:DECIMAL(12,2)|P_COMMENT:VARCHAR(23)"
SUPPLIER_COLUMNS="S_SUPPKEY:DECIMAL(11)|S_NAME:CHAR(25)|S_ADDRESS:VARCHAR(40)|S_NATIONKEY:DECIMAL(11)|S_PHONE:CHAR(15)|S_ACCT
BAL:DECIMAL(12,2)|S_COMMENT:VARCHAR(101)"
PARTSUPP_COLUMNS="PS_PARTKEY:DECIMAL(11)|PS_SUPPKEY:DECIMAL(11)|PS_AVAILQTY:DECIMAL(10)|PS_SUPPLYCOST:DECIMAL(12,2)|PS_COMMEN
T:VARCHAR(199)"
CUSTOMER_COLUMNS="C_CUSTKEY:DECIMAL(11)|C_NAME:VARCHAR(25)|C_ADDRESS:VARCHAR(40)|C_NATIONKEY:DECIMAL(11)|C_PHONE:CHAR(15)|C_A
CCTBAL:DECIMAL(12,2)|C_MKTSEGMENT:CHAR(10)|C_COMMENT:VARCHAR(117)"
ORDERS_COLUMNS="O_ORDERKEY:DECIMAL(11)|O_CUSTKEY:DECIMAL(11)|O_ORDERSTATUS:CHAR(1)|O_TOTALPRICE:DECIMAL(12,2)|O_ORDERDATE:DAT
E|O_ORDERPRIORITY:CHAR(15)|O_CLERK:CHAR(15)|O_SHIPPRIORITY:DECIMAL(10)|O_COMMENT:VARCHAR(79)"
LINEITEM_COLUMNS="L_ORDERKEY:DECIMAL(11)|L_PARTKEY:DECIMAL(11)|L_SUPPKEY:DECIMAL(11)|L_LINENUMBER:DECIMAL(10)|L_QUANTITY:DECI
MAL(12,2)|L_EXTENDEDPRICE:DECIMAL(12,2)|L_DISCOUNT:DECIMAL(12,2)|L_TAX:DECIMAL(12,2)|L_RETURNFLAG:CHAR(1)|L_LINESTATUS:CHAR(1
)|L_SHIPDATE:DATE|L_COMMITDATE:DATE|L_RECEIPTDATE:DATE|L_SHIPINSTRUCT:CHAR(25)|L_SHIPMODE:CHAR(10)|L_COMMENT:VARCHAR(44)"

# since LINEITEM is the largest table by far it is loaded parallel to the other tables
$ParLoader -q -l csv2exa -s $TPC_LOCALDIR/metafile.LINEITEM.csv -d $user:$schema/$pass@$host:$port -a -t LINEITEM -c "`eval
echo \\${LINEITEM_COLUMNS}`" | sed -e "s/\([0-9][0-9]*\)[ \t]*rows loaded\./LINEITEM rows loaded; \1/g" || { echo "Initial
loading failed in table LINEITEM." ; exit 1 ; } &

for table in PART SUPPLIER PARTSUPP CUSTOMER ORDERS
do
    nice $ParLoader -q -l csv2exa -s $TPC_LOCALDIR/metafile.$table.csv -d $user:$schema/$pass@$host:$port -a -t $table -c
"`eval echo \\${${table}_COLUMNS}`" | sed -e "s/\([0-9][0-9]*\)[ \t]*rows loaded\./$table rows loaded; \1/g"  || { echo
"Initial loading failed in table $table." ; exit 1 ; }
done

for table in NATION REGION
do
    nice $Loader -q -l csv2exa -s ${TPC_SINGLENODE_CSVDIR}/Database_SF${SF}/metafile.$table.csv -d
$user:$schema/$pass@$host:$port -a -t $table -c "`eval echo \\${${table}_COLUMNS}`" | sed -e "s/\([0-9][0-9]*\)[ \t]*rows
loaded\./$table rows loaded; \1/g"  || { echo "Initial loading failed in table $table." ; exit 1 ; }
done

# wait until all tables where loaded
wait

sync

exit 0
```

Appendix C: ACID Scripts

```
# ------------------------------------------------------------------------------------------
# tpc_h_load_db_for_acid_tests.sh
# ------------------------------------------------------------------------------------------
#!/bin/bash
# param <nodes> <exadir> <scalefactor> <instances per node> <logdir>

if [ $# -ne 5 ]
then
    echo "Usage: $0 <nodes> <exadir> <scalefactor> <instances per node> <logdir>"
    exit 1
fi

export NODELIST="$1"
export NODES="`echo $NODELIST | tr ',' '\n' | wc -l`"
SPACE_SEP_NODES="`echo -n "$NODES" | tr ',' ' '`"
export ROOTHOST=`echo $NODELIST | sed 's/\([^,]*\),.*/\1/'`
export EXADIR="$2"
export EXABINDIR="$EXADIR/bin"
export LOADERBINDIR="$EXABINDIR"
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$EXADIR/lib
SF="$3"
STREAMS="$4"
INST_PN="$4"
LOGDIR="$5/`date +'%y%m%d%H%M%S'`"
LOGFILE="$LOGDIR/test.log"
ROOTPORT=26822

export TPC_BASEDIR=$HOME/TPCH
export TPC_SCRIPTSDIR=$TPC_BASEDIR/scripts
export TPC_SINGLENODE_CSVDIR="/d02_data/csv/single"
export TPC_SQLDIR=$TPC_BASEDIR/sql
export DSS_CONFIG=$TPC_BASEDIR/tpch_20070105
export TPC_ACID_SQLDIR=$TPC_BASEDIR/ACID/sql

### Startup of database
mkdir -p $LOGDIR
echo "Loading of TPC H database started" >$LOGFILE
echo "-----------------------" >>$LOGFILE
echo "SCRIPT  : $0" >> $LOGFILE
echo "ROOTHOST: $ROOTHOST" >> $LOGFILE
echo "NODES   : $NODELIST" >> $LOGFILE
echo "SCALE   : $SF" >>$LOGFILE
echo "INSTPN  : $INST_PN" >>$LOGFILE
echo "LOGDIR  : $LOGDIR" >>$LOGFILE
echo "SCRIPTS : $TPC_SCRIPTSDIR" >>$LOGFILE
echo "-----------------------" >>$LOGFILE

echo >> $LOGFILE
echo "Start of Test at; `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`" >>$LOGFILE

export EXAPLUS=$EXABINDIR/Console/exaplus

### create user

$EXAPLUS -u sys -P exasol -h $ROOTHOST -p $ROOTPORT -f $TPC_SQLDIR/create_user.sql

###### Phase I (Load-Test)

### Initialization of database-schema

TIMESFILE=$LOGDIR/alltimes.log
LOAD_START="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`"
echo -n "Load Test; $LOAD_START " > $TIMESFILE
echo "Start of Load test at ; $LOAD_START" >> $LOGFILE
$EXAPLUS -p $ROOTPORT -h $ROOTHOST -u tpcuser -P tpcuser -f $TPC_SQLDIR/create_schema.sql >>$LOGDIR/create_schema.log 2>&1

### Load

$TPC_SCRIPTSDIR/load_init_parallel.sh $ROOTHOST $ROOTPORT tpcuser tpcuser tpc $SF $INST_PN >>$LOGFILE 2>&1
sync

### Initialization of indices
echo "Start of database optimization at ; `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`" >> $LOGFILE
optimizeLineitem()
{
    cat $TPC_SQLDIR/create_indices.sql | grep -i -v -e NATION -e REGION -e " PART" -e ORDERS -e PARTSUPP -e CUSTOMER -e
SUPPLIER -e ANALYZE | $EXAPLUS -p $ROOTPORT -h $ROOTHOST -u tpcuser -P tpcuser 1>>$LOGDIR/optimizeLineitem.log 2>&1
}

optimizeOther()
{
    cat $TPC_SQLDIR/create_indices.sql | grep -i -v -e LINEITEM -e ANALYZE | $EXAPLUS -p $ROOTPORT -h $ROOTHOST -u tpcuser -P
tpcuser 1>>$LOGFILE 2>&1
}

optimizeLineitem &
optimizeOther &
wait

cat $LOGDIR/optimizeLineitem.log >> $LOGFILE
rm -f $LOGDIR/optimizeLineitem.log

cat $TPC_SQLDIR/create_indices.sql | grep -i -v -e ENFORCE -e OPTIMIZE -e SELECT | $EXAPLUS -p $ROOTPORT -h $ROOTHOST -u
tpcuser -P tpcuser 1>>$LOGFILE 2>&1
```

```
echo "End of database optimization at ; `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`" >> $LOGFILE
sync

LOAD_END="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`"
echo "End of Load test at ; $LOAD_END" >> $LOGFILE
echo -n "; $LOAD_END " >> $TIMESFILE
LOAD_TIME="""`$TPC_SCRIPTSDIR/timediff "$LOAD_START" "$LOAD_END"`"""
echo "; $LOAD_TIME" >> $TIMESFILE
sync

# execute tables.sql
cat $TPC_SQLDIR/tables.sql | $EXAPLUS -p $ROOTPORT -h $ROOTHOST -u tpcuser -P tpcuser >>$LOGDIR/tables.log 2>&1

# execute dbcheck.sql
cat $TPC_SQLDIR/dbcheck.sql | $EXAPLUS -p $ROOTPORT -h $ROOTHOST -u tpcuser -P tpcuser >>$LOGDIR/dbcheck.log 2>&1

# create history table
cat $TPC_ACID_SQLDIR/create_history_table.sql | $EXAPLUS -p $ROOTPORT -h $ROOTHOST -u tpcuser -P tpcuser >>$LOGFILE 2>&1

echo "End of Test at ; " `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >>$LOGFILE



# ----------------------------------------------------------------------------------------
# run_acid.sh
# ----------------------------------------------------------------------------------------
#!/bin/bash
# params <roothost> <scalefactor> <streams> <logdir> <durability_tests>

if [ ! $# -eq 5 ]
then
    echo "usage: $0 <roothost> <scalefactor> <streams> <logdir> <durability_tests>"
    exit 1
fi

ROOTHOST="$1"
SCALEFACTOR="$2"
STREAMS="$3"
LOGDIR="$4/ACID"
NUM_DURABILITY="$5"

ROOTPORT=26822

mkdir -p $LOGDIR

ACIDDIR="$HOME/TPCH/ACID"
TMPDIR=$LOGDIR/tmp
mkdir -p $TMPDIR

echo "TPC Benchmark H - ACID-Test" | tee $LOGDIR/ACID.log
echo "===========================" | tee -a $LOGDIR/ACID.log
echo | tee -a $LOGDIR/ACID.log

## create the history table
echo "creating history table" | tee -a $LOGDIR/ACID.log
# $EXAPLUS -h $ROOTHOST -p $ROOTPORT -u sys -P exasol -f $ACIDDIR/sql/create_history_table.sql >> /dev/null
echo | tee -a $LOGDIR/ACID.log

## ATOMICITY TEST
A_TRANSACTIONS_COMMIT=25
A_TRANSACTIONS_ROLLBACK=25

$ACIDDIR/scripts/run_atomicity_test.sh $ROOTHOST $SCALEFACTOR $LOGDIR $A_TRANSACTIONS_COMMIT $A_TRANSACTIONS_ROLLBACK | tee -
a $LOGDIR/ACID.log

## CONSISTENCY TEST
# $ACIDDIR/scripts/run_consistency_test.sh $ROOTHOST $SCALEFACTOR $STREAMS $LOGDIR  | tee -a $LOGDIR/ACID.log


## ISOLATION TEST
$ACIDDIR/scripts/run_isolation_test.sh $ROOTHOST $SCALEFACTOR $LOGDIR | tee -a $LOGDIR/ACID.log

## DURABILITY TEST
$ACIDDIR/scripts/run_durability_test.sh $ROOTHOST $SCALEFACTOR $STREAMS $LOGDIR $NUM_DURABILITY | tee -a $LOGDIR/ACID.log

echo | tee -a $LOGDIR/ACID.log

echo "====================" | tee -a $LOGDIR/ACID.log
echo "Testresult ACID Test" | tee -a $LOGDIR/ACID.log
echo "====================" | tee -a $LOGDIR/ACID.log

if [ -z "`grep 'failed' $LOGDIR/ACID.log`" ]
then
        echo "ACID test passed" | tee -a $LOGDIR/ACID.log
else
        echo "ACID test failed" | tee -a $LOGDIR/ACID.log
fi



# ----------------------------------------------------------------------------------------
# run_atomicity_test.sh
# ----------------------------------------------------------------------------------------
#!/bin/bash
# params <roothost> <scalefactor> <logdir> <transactionsForTest1> <transactionsForTest2>

ROOTHOST="$1"
SCALEFACTOR="$2"
LOGDIR="$3/atomicity"
```

```
TRANSACTIONSCOMMIT="$4"
TRANSACTIONSROLLBACK="$5"

mkdir -p $LOGDIR
mkdir -p $LOGDIR/tmp
ACIDDIR="$HOME/TPCH/ACID"
LOGFILE=$LOGDIR/atomicity.log

echo "**************" | tee $LOGFILE
echo "ATOMICITY_TEST" | tee -a $LOGFILE
echo "**************" | tee -a $LOGFILE

$ACIDDIR/scripts/atomicity_test_1.sh $ROOTHOST $SCALEFACTOR $LOGDIR $TRANSACTIONSCOMMIT | tee -a $LOGFILE
$ACIDDIR/scripts/atomicity_test_2.sh $ROOTHOST $SCALEFACTOR $LOGDIR $TRANSACTIONSROLLBACK | tee -a $LOGFILE

if [ -z "`grep 'failed' $LOGFILE`" ]
then
        echo "Atomicity test passed" | tee -a $LOGFILE
else
        echo "Atomicity test failed" | tee -a $LOGFILE
fi

echo | tee -a $LOGFILE



# ------------------------------------------------------------------------------------
# atomicity_test_1.sh
# ------------------------------------------------------------------------------------
#!/bin/bash
# executes atomicity test 1
# param: <roothost> <scalefactor> <logdir> <numberOfTransactions>

ROOTHOST="$1"
SCALEFACTOR="$2"
LOGDIR="$3"
NUMUPDATES="$4"

ACIDDIR="$HOME/TPCH/ACID"

$ACIDDIR/scripts/atomicity_test_mainpart.sh COMMIT $ROOTHOST $SCALEFACTOR $LOGDIR 1 $NUMUPDATES
LINESCHANGED=(`wc -l $LOGDIR/diff_atomicity_1.diff`)

if [ $LINESCHANGED -eq $((4+1+$NUMUPDATES+4+(4*$NUMUPDATES)+2+(2*$NUMUPDATES) )) ]
then
        echo "Atomicity Test 1 passed ($LINESCHANGED differences in diff file)"
else
        if [ ! $LINESCHANGED -eq 4 ]
        then
                echo "Atomicity Test 1 resulted in changed lines (might have failed), but diff file contains different
number of lines ($LINESCHANGED) then expected ($((4+1+$NUMUPDATES+4+(4*$NUMUPDATES)+2+(2*$NUMUPDATES))) ). Check diff file,
please, since test might have succeeded."
        else
                echo "Atomicity Test 1 failed ($LINESCHANGED differences in diff file which very probably means no lines
where changed)."
        fi
fi



# ------------------------------------------------------------------------------------
# atomicity_test_2.sh
# ------------------------------------------------------------------------------------
#!/bin/bash
# executes atomicity test 2
# param: <roothost> <scalefactor> <logdir> <numberOfTransactions>

ROOTHOST="$1"
SCALEFACTOR="$2"
LOGDIR="$3"
NUMUPDATES="$4"

ACIDDIR="$HOME/TPCH/ACID"

$ACIDDIR/scripts/atomicity_test_mainpart.sh ROLLBACK $ROOTHOST $SCALEFACTOR $LOGDIR 2 $NUMUPDATES
LINESCHANGED=(`wc -l $LOGDIR/diff_atomicity_2.diff`)

if [ $LINESCHANGED -eq 4 ]
then
        echo "Atomicity Test 2 passed ($LINESCHANGED differences in diff file)"
else
        echo "Atomicity Test 2 failed ($LINESCHANGED differences in diff file which very probably means some lines where
changed)."
fi



# ------------------------------------------------------------------------------------
# atomicity_test_mainpart.sh
# ------------------------------------------------------------------------------------
#!/bin/bash
# param <commit or rollback> <host> <scalefactor> <logdir> <acidnumber> <numUpdates>

ACTION="$1"
ROOTHOST="$2"
SCALEFACTOR="$3"
LOGDIR="$4"
ACIDNUMBER="$5"
```

```
NUMUPDATES="$6"

ROOTPORT=26822

LOGFILE=$LOGDIR/atomicity_debug.log

ACIDDIR="$HOME/TPCH/ACID"

# generate numUpdates random orderkeys
$ACIDDIR/scripts/create_random_transactions.sh       0       $NUMUPDATES       $SCALEFACTOR       $ROOTHOST       $LOGDIR/tmp
$LOGDIR/orderkeys_$ACIDNUMBER.log

# save old results for that key values
$ACIDDIR/scripts/store_values_for_orderkeys.sh                  $ROOTHOST                  $LOGDIR/orderkeys_$ACIDNUMBER.log
$LOGDIR/before_atomicity_$ACIDNUMBER.log $LOGDIR

# execute all transactions
$ACIDDIR/scripts/transaction/acidtransaction    $ROOTHOST    $ROOTPORT    $ACTION    NOWAIT    $LOGDIR/orderkeys_$ACIDNUMBER.log
$LOGDIR/atomicity${ACIDNUMBER}_return_transaction.log tpcuser tpcuser $LOGDIR/atomicity${ACIDNUMBER}_values.log >> $LOGFILE

# save new results for that key values
$ACIDDIR/scripts/store_values_for_orderkeys.sh                  $ROOTHOST                  $LOGDIR/orderkeys_$ACIDNUMBER.log
$LOGDIR/after_atomicity_$ACIDNUMBER.log $LOGDIR

# compare old and new files
diff             $LOGDIR/before_atomicity_$ACIDNUMBER.log             $LOGDIR/after_atomicity_$ACIDNUMBER.log             >
$LOGDIR/diff_atomicity_$ACIDNUMBER.diff



# ------------------------------------------------------------------------------------
# run_isolation_test.sh
# ------------------------------------------------------------------------------------
#!/bin/bash
# params: <host> <scalefactor> <logdir>

HOST="$1"
SF="$2"
LOGDIR="$3/isolation"
mkdir -p $LOGDIR

ACIDDIR="$HOME/TPCH/ACID"
TMPDIR=$LOGDIR/tmp
mkdir -p $TMPDIR

echo "**************" | tee $LOGDIR/isolation.log
echo "ISOLATION TEST" | tee -a $LOGDIR/isolation.log
echo "**************" | tee -a $LOGDIR/isolation.log

## ISOLATION TEST 1
# generate transaction for isolation test 1
$ACIDDIR/scripts/create_random_transactions.sh 0 1 $SF $HOST $TMPDIR $LOGDIR/transaction_keys_isolation1.keys
# execute isolation test 1
$ACIDDIR/scripts/isolation1.sh $LOGDIR/transaction_keys_isolation1.keys $HOST $LOGDIR | tee -a $LOGDIR/isolation.log

## ISOLATION TEST 2
# generate transaction for isolation test 2
$ACIDDIR/scripts/create_random_transactions.sh 0 1 $SF $HOST $TMPDIR $LOGDIR/transaction_keys_isolation2.keys
# execute isolation test 2
$ACIDDIR/scripts/isolation2.sh $LOGDIR/transaction_keys_isolation2.keys $HOST $LOGDIR  | tee -a $LOGDIR/isolation.log

## ISOLATION TEST 3
# generate transaction for isolation test 3
 $ACIDDIR/scripts/create_random_transactions.sh 0 1 $SF $HOST $TMPDIR $LOGDIR/transaction_keys_isolation3.keys
# execute isolation test 3
 $ACIDDIR/scripts/isolation3.sh $LOGDIR/transaction_keys_isolation3.keys $HOST $LOGDIR  | tee -a $LOGDIR/isolation.log

## ISOLATION TEST 4
# generate transaction for isolation test 4
 $ACIDDIR/scripts/create_random_transactions.sh 0 1 $SF $HOST $TMPDIR $LOGDIR/transaction_keys_isolation4.keys
# execute isolation test 4
 $ACIDDIR/scripts/isolation4.sh $LOGDIR/transaction_keys_isolation4.keys $HOST $LOGDIR  | tee -a $LOGDIR/isolation.log

## ISOLATION TEST 5
# generate transaction for isolation test 5
$ACIDDIR/scripts/create_random_transactions.sh 0 1 $SF $HOST $TMPDIR $LOGDIR/transaction_keys_isolation5.keys
# execute isolation test 5
$ACIDDIR/scripts/isolation5.sh $LOGDIR/transaction_keys_isolation5.keys $HOST $LOGDIR  | tee -a $LOGDIR/isolation.log

## ISOLATION TEST 6
# generate transaction for isolation test 6
$ACIDDIR/scripts/create_random_transactions.sh 0 1 $SF $HOST $TMPDIR $LOGDIR/transaction_keys_isolation6.keys
# execute isolation test 6
$ACIDDIR/scripts/isolation6.sh $LOGDIR/transaction_keys_isolation6.keys $HOST $LOGDIR  | tee -a $LOGDIR/isolation.log

## determine test result
if [ -z "`grep 'failed' $LOGDIR/isolation.log`" ]
then
        echo "Isolation test passed"
else
        echo "Isolation test failed. See $LOGDIR/isolation.log for details."
fi



# ------------------------------------------------------------------------------------
# isolation1.sh
# ------------------------------------------------------------------------------------
#!/bin/bash
```

```
# params: <inputfile> <host> <logdir> <keypress>

INPUTFILE="$1"
HOST="$2"
LOGDIR="$3"
KEYPRESS="$4"

ROOTPORT=26822

OKEY=( `cat $INPUTFILE` )
SCRIPTSDIR="$HOME/TPCH/ACID/scripts"
LOGFILE=$LOGDIR/isolation1.log

PIPENAME=$LOGDIR/tmp/i1pipe

mkfifo $PIPENAME

echo "----------------" | tee $LOGFILE
echo "Isolation test 1" | tee -a $LOGFILE
echo "----------------" | tee -a $LOGFILE
echo >> $LOGFILE
echo -n "O_KEY, L_KEY, DELTA = " >> $LOGFILE
cat $INPUTFILE >> $LOGFILE
echo >> $LOGFILE

# Step 1: start ACID query
echo "Step 1: ACID query" | tee -a $LOGFILE
echo Before query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
RESULTS1="`$SCRIPTSDIR/acidquery.sh $HOST $LOGDIR $OKEY | tee -a $LOGFILE`"
echo After query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
# Step 2: start update transaction on same row, wait before commit
# acidtransaction with wait and commit
if [ ! -z $KEYPRESS ]
then
         echo "Press enter to start transaction (Step 2)"
         read $X
fi

echo "Step 2: ACID transaction" | tee -a $LOGFILE
cat    $PIPENAME    |    $SCRIPTSDIR/transaction_isolation/acidtransaction    $HOST    $ROOTPORT    COMMIT    WAIT    $INPUTFILE
$LOGDIR/isolation1_return_t1.log tpcuser tpcuser 2>&1 | sed -u "s/^/Transaction 1:/g" | tee -a $LOGFILE  | grep -A 1 "ERROR"
&
# wait for transaction to reach wait for commit
sleep 10

# Step 3: start ACID query again, verify results are still the old
if [ ! -z $KEYPRESS ]
then
         echo "Press enter to start ACID query again (Step 3)"
         read $X
fi
echo "Step 3: Same ACID query again" | tee -a $LOGFILE
echo Before query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
RESULTS2="`$SCRIPTSDIR/acidquery.sh $HOST $LOGDIR $OKEY | tee -a $LOGFILE`"
echo After query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
if [ "$RESULTS1" != "$RESULTS2" ]
then
         echo "Failed: Results differ from results of first query." | tee -a $LOGFILE
         TESTRESULT="Test failed"
fi

# Step 4: commit the update transaction
if [ ! -z $KEYPRESS ]
then
         echo "Press enter to commit ACID transaction (Step 4)"
         read $X
fi
echo "Step 4: Commit of ACID transaction" | tee -a $LOGFILE
# send newline to acidtransaction 1 (to commit)
FD=8
eval "exec $FD>>$PIPENAME" # open pipe
echo >& $FD
eval "exec $FD>>-" >/dev/null 2>&1  # close pipe (EOF)
# wait for commit to finish
sleep 10

# Step 5: start ACID query a third time and verify that the new row is retrieved
if [ ! -z $KEYPRESS ]
then
         echo "Press enter to start ACID query a third time (Step 5)"
         read $X
fi
echo "Step 5: Same ACID query again" | tee -a $LOGFILE
echo Before query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
RESULTS3="`$SCRIPTSDIR/acidquery.sh $HOST $LOGDIR $OKEY | tee -a $LOGFILE`"
echo After query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
if [ "$RESULTS2" = "$RESULTS3" ]
then
         echo "Failed: Results are the same as those from the second query." | tee -a $LOGFILE
         TESTRESULT="Test failed"
fi

ERRORS="`grep 'ERROR' $LOGFILE`"
if [ \( -z "$TESTRESULT" \) -a \( -z "$ERRORS" \) ]
then
         echo "Isolation test 1 passed" | tee -a $LOGFILE
else
         echo "Isolation test 1 failed" | tee -a $LOGFILE
fi

rm -f $PIPENAME
```

```
# ------------------------------------------------------------------------------------------
# isolation2.sh
# ------------------------------------------------------------------------------------------
#!/bin/bash
# params: <inputfile> <host> <logdir> <keypress>

INPUTFILE="$1"
HOST="$2"
LOGDIR="$3"
KEYPRESS="$4"

ROOTPORT=26822

OKEY=( `cat $INPUTFILE` )
SCRIPTSDIR="$HOME/TPCH/ACID/scripts"
LOGFILE=$LOGDIR/isolation2.log

PIPENAME=$LOGDIR/tmp/i2pipe

mkfifo $PIPENAME

echo "----------------" | tee $LOGFILE
echo "Isolation test 2" | tee -a $LOGFILE
echo "----------------" | tee -a $LOGFILE
echo >> $LOGFILE
echo -n "O_KEY, L_KEY, DELTA = " >> $LOGFILE
cat $INPUTFILE >> $LOGFILE
echo >> $LOGFILE

# Step 1: start ACID query
echo "Step 1: ACID query" | tee -a $LOGFILE
echo Before query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
RESULTS1="`$SCRIPTSDIR/acidquery.sh $HOST $LOGDIR $OKEY | tee -a $LOGFILE`"
echo After query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE

# Step 2: start update transaction on same row, wait before commit
# acidtransaction with wait and commit
if [ ! -z $KEYPRESS ]
then
         echo "Press enter to start transaction (Step 2)"
         read $X
fi

echo "Step 2: ACID transaction" | tee -a $LOGFILE
cat   $PIPENAME   |   $SCRIPTSDIR/transaction_isolation/acidtransaction   $HOST   $ROOTPORT   ROLLBACK   WAIT   $INPUTFILE
$LOGDIR/isolation2_return_t1.log tpcuser tpcuser 2>&1 | sed -u 's/^/Transaction 1:/g' | tee -a $LOGFILE | grep -A 1 "ERROR" &
# wait for transaction to reach wait for commit
sleep 10

# Step 3: start ACID query again, verify results are still the old
if [ ! -z $KEYPRESS ]
then
         echo "Press enter to start ACID query again (Step 3)"
         read $X
fi
echo "Step 3: Same ACID query again" | tee -a $LOGFILE
echo Before query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
RESULTS2="`$SCRIPTSDIR/acidquery.sh $HOST $LOGDIR $OKEY | tee -a $LOGFILE`"
echo After query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
if [ "$RESULTS1" != "$RESULTS2" ]
then
         echo "Failed: Results differ from results of first query." | tee -a $LOGFILE
         TESTRESULT="Test failed"
fi

# Step 4: rollback the update transaction
if [ ! -z $KEYPRESS ]
then
         echo "Press enter to rollback ACID transaction (Step 4)"
         read $X
fi
echo "Step 4: Rollback of ACID transaction" | tee -a $LOGFILE
# send newline to acidtransaction 1 (to rollback)
FD=8
eval "exec $FD>>$PIPENAME" # open pipe
echo >& $FD
eval "exec $FD>>-" >/dev/null 2>&1  # close pipe (EOF)
# wait for commit to finish
sleep 10

# Step 5: start ACID query a third time and verify that the new row is retrieved
if [ ! -z $KEYPRESS ]
then
         echo "Press enter to start ACID query a third time (Step 5)"
         read $X
fi
echo "Step 5: Same ACID query again" | tee -a $LOGFILE
echo Before query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
RESULTS3="`$SCRIPTSDIR/acidquery.sh $HOST $LOGDIR $OKEY | tee -a $LOGFILE`"
echo After query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
if [ "$RESULTS2" != "$RESULTS3" ]
then
         echo "Failed: Results differ from those of the second query." | tee -a $LOGFILE
         TESTRESULT="Test failed"
fi
```

```
# determine test result
ERRORS="grep 'ERROR' $LOGFILE"
if [ \( -z "$TESTRESULT" \) -a \( ! -z "$ERRORS" \) ]
then
         echo "Isolation test 2 passed" | tee -a $LOGFILE
else
         echo "Isolation test 2 failed" | tee -a $LOGFILE
fi

rm -f $PIPENAME



# ----------------------------------------------------------------------------------------
# isolation3.sh
# ----------------------------------------------------------------------------------------
#!/bin/bash
# params: <inputfile> <host> <logdir> <keypress>

INPUTFILE="$1"
HOST="$2"
LOGDIR="$3"
KEYPRESS="$4"

ROOTPORT=26822

OKEY=( `cat $INPUTFILE` )
SCRIPTSDIR="$HOME/TPCH/ACID/scripts"
LOGFILE=$LOGDIR/isolation3.log

PIPENAME=$LOGDIR/tmp/i3pipe

mkfifo $PIPENAME

echo "----------------" | tee $LOGFILE
echo "Isolation test 3" | tee -a $LOGFILE
echo "----------------" | tee -a $LOGFILE
echo >> $LOGFILE
echo -n "O_KEY, L_KEY, DELTA = " >> $LOGFILE
cat $INPUTFILE >> $LOGFILE
echo >> $LOGFILE

# Step 1: start a query and verify that the row was retrieved
echo "Step 1: ACID query" | tee -a $LOGFILE
echo Before query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
RESULTS1="`$SCRIPTSDIR/acidquery.sh $HOST $LOGDIR $OKEY | tee -a $LOGFILE`"
echo After query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE

# Step 2: start update transation, wait before commit
if [ ! -z "$KEYPRESS" ]
then
         echo "Press enter to start ACID transaction 1 (Step 2)"
         read $X
fi
echo "Step 2: ACID transaction 1 until commit" | tee -a $LOGFILE
cat $PIPENAME | $SCRIPTSDIR/transaction_isolation/acidtransaction $HOST $ROOTPORT COMMIT WAIT $INPUTFILE
$LOGDIR/isolation3_return_t1.log tpcuser tpcuser | sed -u 's/^/Transaction 1:/g' | tee -a $LOGFILE | grep -A 1
"\(ERROR\|ROLLBACK\)" &
sleep 10
if [ ! -z "`grep '\(ERROR\|ROLLBACK\)' $LOGFILE`" ]
then
         ERROR="$ERROR Transaction 1"
fi

# Step 3: start another update transaction on same row. Verify rollback
if [ ! -z "$KEYPRESS" ]
then
         echo "Press enter to start ACID transaction 2 (Step 3)"
         read $X
fi
echo "Step 3: ACID transaction 2 with COMMIT (expecting ROLLBACK exception)" | tee -a $LOGFILE
$SCRIPTSDIR/transaction_isolation/acidtransaction $HOST $ROOTPORT COMMIT NOWAIT $INPUTFILE $LOGDIR/isolation3_return_t2.log
tpcuser tpcuser | sed 's/^/Transaction 2:/g' | tee -a $LOGFILE | grep -A 1 "ROLLBACK" | tee
$LOGDIR/isolation3_rollback_t2.log
ROLLBACK="`grep 'ROLLBACK' $LOGDIR/isolation3_rollback_t2.log`"
if [ -z "$ROLLBACK" ]
then
     ERROR="$ERROR No Rollback in Transaction 2"
fi

if [ ! -z "`grep 'ERROR' $LOGFILE`" ]
then
         ERROR="$ERROR Transaction 2"
fi

# Step 4: commit the update transaction
if [ ! -z "$KEYPRESS" ]
then
         echo "Press enter to commit ACID transaction 1 (Step 4)"
         read $X
fi
echo "Step 4: Commit of ACID transaction 1" | tee -a $LOGFILE
# send newline to acidtransaction 1 (to commit)
FD=8
eval "exec $FD>>$PIPENAME" # open pipe
echo >& $FD
eval "exec $FD>>-" >/dev/null 2>&1  # close pipe (EOF)
sleep 10
if [ ! -z "`grep '\(ERROR\|ROLLBACK\)' $LOGFILE | grep 'Transaction 1'`" ]
then
```

```
        ERROR="$ERROR Transaction 1"
fi

# Step 5: start the same query again and verify that the new row was retrieved
if [ ! -z "$KEYPRESS" ]
then
        echo "Press enter to start ACID query again (Step 5)"
        read $X
fi
echo "Step 5: ACID query again" | tee -a $LOGFILE
echo Before query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
RESULTS2="`$SCRIPTSDIR/acidquery.sh $HOST $LOGDIR $OKEY | tee -a $LOGFILE`"
echo After query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
if [ "$RESULTS1" = "$RESULTS2" ]
then
        ERROR="$ERROR Results compare"
fi

# determine test result
if [ ! -z "$ERROR" ]
then
        echo "Isolation test 3 failed." | tee -a $LOGFILE
        echo "$ERROR"
else
        echo "Isolation test 3 passed." | tee -a $LOGFILE
fi

rm -f $PIPENAME



# ------------------------------------------------------------------------------------
# isolation4.sh
# ------------------------------------------------------------------------------------
#!/bin/bash
# params: <inputfile> <host> <logdir> <keypress>

INPUTFILE="$1"
HOST="$2"
LOGDIR="$3"
KEYPRESS="$4"

ROOTPORT=26822

OKEY=( `cat $INPUTFILE` )
SCRIPTSDIR="$HOME/TPCH/ACID/scripts"
LOGFILE="$LOGDIR/isolation4.log"

PIPENAME=$LOGDIR/tmp/i4pipe

mkfifo $PIPENAME

echo "----------------" | tee $LOGFILE
echo "Isolation test 4" | tee -a $LOGFILE
echo "----------------" | tee -a $LOGFILE
echo >> $LOGFILE
echo -n "O_KEY, L_KEY, DELTA = " >> $LOGFILE
cat $INPUTFILE >> $LOGFILE
echo >> $LOGFILE

# Step 1: start a query and verify that the row was retrieved
echo "Step 1: ACID query" | tee -a $LOGFILE
echo Before query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
RESULTS1="`$SCRIPTSDIR/acidquery.sh $HOST $LOGDIR $OKEY | tee -a $LOGFILE`"
echo After query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE

# Step 2: start update transation, wait before commit
if [ ! -z "$KEYPRESS" ]
then
        echo "Press enter to start ACID transaction 1 (Step 2)"
        read $X
fi
echo "Step 2: ACID transaction 1 until commit" | tee -a $LOGFILE
cat $PIPENAME | $SCRIPTSDIR/transaction_isolation/acidtransaction $HOST $ROOTPORT ROLLBACK WAIT $INPUTFILE
$LOGDIR/isolation4_return_t1.log tpcuser tpcuser | sed -u 's/^/Transaction 1:/g' | tee -a $LOGFILE | grep -A 1 "ERROR" &
sleep 10
if [ ! -z "`grep '\(ERROR\|ROLLBACK\)' $LOGFILE`" ]
then
        ERROR="$ERROR Transaction 1"
fi

# Step 3: start another update transaction on same row. Verify rollback
if [ ! -z "$KEYPRESS" ]
then
        echo "Press enter to start ACID transaction 2 (Step 3)"
        read $X
fi
echo "Step 3: ACID transaction 2 with COMMIT (expecting ROLLBACK exception)" | tee -a $LOGFILE
$SCRIPTSDIR/transaction_isolation/acidtransaction $HOST $ROOTPORT COMMIT NOWAIT $INPUTFILE $LOGDIR/isolation4_return_t2.log
tpcuser tpcuser | sed -u 's/^/Transaction 2:/g' | tee -a $LOGFILE | grep -A 1 "ROLLBACK" | tee
$LOGDIR/isolation4_rollback_t2.log
ROLLBACK="`grep 'ROLLBACK' $LOGDIR/isolation4_rollback_t2.log`"
if [ -z "$ROLLBACK" ]
then
        ERROR="$ERROR No Rollback in Transaction 2"
fi
if [ ! -z "`grep 'ERROR' $LOGFILE`" ]
then
        ERROR="$ERROR Transaction 2"
fi
```

```
# Step 4: commit the update transaction
if [ ! -z "$KEYPRESS" ]
then
          echo "Press enter to rollback ACID transaction 1 (Step 4)"
          read $X
fi
echo "Step 4: Rollback of ACID transaction 1" | tee -a $LOGFILE
# send newline to acidtransaction 1 (to rollback)
FD=8
eval "exec $FD>>$PIPENAME" # open pipe
echo >& $FD
eval "exec $FD>>-" >/dev/null 2>&1  # close pipe (EOF)
sleep 10
if [ ! -z "`grep 'ERROR' $LOGFILE | grep 'Transaction 1'`" ]
then
          ERROR="$ERROR Transaction 1"
fi

# Step 5: start the same query again and verify that the new row was retrieved
if [ ! -z "$KEYPRESS" ]
then
          echo "Press enter to start ACID query again (Step 5)"
          read $X
fi
echo "Step 5: ACID query again" | tee -a $LOGFILE
echo Before query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
RESULTS2="`$SCRIPTSDIR/acidquery.sh $HOST $LOGDIR $OKEY | tee -a $LOGFILE`"
echo After query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
if [ "$RESULTS1" != "$RESULTS2" ]
then
          ERROR="$ERROR Results compare"
fi

## determine test result
if [ ! -z "$ERROR" ]
then
          echo "Isolation test 4 failed." | tee -a $LOGFILE
          echo "$ERROR"
else
          echo "Isolation test 4 passed." | tee -a $LOGFILE
fi

rm -f $PIPENAME



# ------------------------------------------------------------------------------------------
# isolation5.sh
# ------------------------------------------------------------------------------------------
#!/bin/bash
# params: <inputfile> <host> <logdir>

INPUTFILE="$1"
HOST="$2"
LOGDIR="$3"

ROOTPORT=26822

OKEY=( `cat $INPUTFILE` )
SCRIPTSDIR="$HOME/TPCH/ACID/scripts"
LOGFILE=$LOGDIR/isolation5.log

PIPENAME=$LOGDIR/tmp/i5pipe

mkfifo $PIPENAME

echo "----------------" | tee $LOGFILE
echo "Isolation test 5" | tee -a $LOGFILE
echo "----------------" | tee -a $LOGFILE
echo >> $LOGFILE
echo -n "O_KEY, L_KEY, DELTA = " >> $LOGFILE
cat $INPUTFILE >> $LOGFILE
echo >> $LOGFILE

echo "Copying lineitem to lineitem2, orders to orders2, history to history2" | tee -a $LOGFILE

# before: Copying lineitem to lineitem2
#         Copying orders to orders2
#         Copying history to history2
$SCRIPTSDIR/copyTablesForIsolation5.sh $HOST $LOGDIR/tmp >> $LOGFILE


# Step 1: Start query against first table
echo "Step 1: ACID query on LINEITEM" | tee -a $LOGFILE
echo Before query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
RESULT1Q1="`$SCRIPTSDIR/acidquery.sh $HOST $LOGDIR $OKEY | tee -a $LOGFILE`"
echo After query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE

# Step 2: Start query against second table
echo "Step 2: ACID query on LINEITEM2" | tee -a $LOGFILE
echo Before query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
RESULT1Q2="`$SCRIPTSDIR/acidquery.sh $HOST $LOGDIR $OKEY LINEITEM2 | tee -a $LOGFILE`"
echo After query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE

# Step 3: Start update transaction against first tables, wait before commit
if [ ! -z "$KEYPRESS" ]
then
          echo "Press enter to start ACID transaction 1 (on LINEITEM, ORDERS) (Step 3)"
          read $X
```

```
fi
echo "Step 3: ACID transaction 1 (on LINEITEM, ORDERS) until commit" | tee -a $LOGFILE
cat    $PIPENAME    |    $SCRIPTSDIR/transaction_isolation/acidtransaction    $HOST    $ROOTPORT    COMMIT    WAIT    $INPUTFILE
$LOGDIR/isolation5_return_t1.log  tpcuser  tpcuser  |  sed  -u  's/^/Transaction  1:/g'  |  tee  -a  $LOGFILE  |  grep  -A  1
"\(ROLLBACK\|ERROR\)" &
sleep 10
if [ ! -z "`grep '\(ROLLBACK\|ERROR\)' $LOGFILE`" ]
then
         ERROR="$ERROR Transaction1"
fi

# Step 4: Start update transaction against second tables and commit
if [ ! -z "$KEYPRESS" ]
then
         echo "Press enter to start ACID transaction 2 (on LINEITEM2, ORDERS2) (Step 4)"
         read $X
fi
echo "Step 4: ACID transaction 2 (on LINEITEM2, ORDERS2) with COMMIT" | tee -a $LOGFILE
$SCRIPTSDIR/transaction_isolation/acidtransaction $HOST $ROOTPORT COMMIT NOWAIT $INPUTFILE $LOGDIR/isolation5_return_t2.log
tpcuser  tpcuser  lineitem2  orders2  history2  |  sed  -u  's/^/Transaction  2:/g'  |  tee  -a  $LOGFILE  |  grep  -A  1
"\(ROLLBACK\|ERROR\)"
if [ ! -z "`grep '\(ROLLBACK\|ERROR\)' $LOGFILE | grep 'Transaction 2'`" ]
then
         ERROR="$ERROR Transaction2"
fi

# Step 5: Commit first update transaction
if [ ! -z "$KEYPRESS" ]
then
         echo "Press enter to commit ACID transaction 1 (Step 5)"
         read $X
fi
echo "Step 5: Commit of ACID transaction 1" | tee -a $LOGFILE
# send newline to acidtransaction 1 (to rollback)
FD=8
eval "exec $FD>>$PIPENAME" # open pipe
echo >& $FD
eval "exec $FD>>-" >/dev/null 2>&1  # close pipe (EOF)
sleep 10
if [ ! -z "`grep '\(ROLLBACK\|ERROR\)' $LOGFILE | grep 'Transaction 1'`" ]
then
         ERROR="$ERROR Transaction1"
fi

# Step 6: Start query against first table again (results should differ from first time)
if [ ! -z "$KEYPRESS" ]
then
         echo "Press enter to start ACID query on LINEITEM again (Step 6)"
         read $X
fi
echo "Step 6: ACID query again (on LINEITEM)" | tee -a $LOGFILE
echo Before query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
RESULT2Q1="`$SCRIPTSDIR/acidquery.sh $HOST $LOGDIR $OKEY | tee -a $LOGFILE`"
echo After query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
if [ "$RESULT1Q1" = "$RESULT2Q1" ]
then
         ERROR="$ERROR Compare results for first table"
fi

# Step 7: Start query against second table again (results should differ from first time)
if [ ! -z "$KEYPRESS" ]
then
         echo "Press enter to start ACID query on LINEITEM2 again (Step 7)"
         read $X
fi
echo "Step 7: ACID query again (on LINEITEM2)" | tee -a $LOGFILE
echo Before query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
RESULT2Q2="`$SCRIPTSDIR/acidquery.sh $HOST $LOGDIR $OKEY LINEITEM2 | tee -a $LOGFILE`"
echo After query `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'` >> $LOGFILE
if [ "$RESULT1Q2" = "$RESULT2Q2" ]
then
         ERROR="$ERROR Compare results for second table"
fi

if [ -z "$ERROR" ]
then
         echo "Isolation test 5 passed"
else
         echo "Isolation test 5 failed"
         echo $ERROR
fi

echo "Deleting lineitem2, orders2, history2" | tee -a $LOGFILE
$SCRIPTSDIR/deleteCopiedTablesForIsolation5.sh $HOST $LOGDIR/tmp >> $LOGFILE

rm -f $PIPENAME



# ------------------------------------------------------------------------------------
# copyTablesForIsolation5.sh
# ------------------------------------------------------------------------------------
#!/bin/bash
# params: <HOST> <TMPDIR>

HOST=$1
TMPDIR=$2

ROOTPORT=26822
```

```
TMPFILE=$TMPDIR/tmp_copy.sql

echo "set autocommit off;" > $TMPFILE
echo "open schema tpc;" > $TMPFILE
echo "create table LINEITEM2 as select * from lineitem;" >> $TMPFILE
echo "create table ORDERS2 as select * from orders;" >> $TMPFILE
echo "create table HISTORY2 as select * from history;" >> $TMPFILE
echo "commit;" >> $TMPFILE

$EXAPLUS -h $HOST -p $ROOTPORT -u tpcuser -P tpcuser -f $TMPFILE
rm -f $TMPFILE




# ------------------------------------------------------------------------------------------
# deleteCopiedTablesForIsolation5.sh
# ------------------------------------------------------------------------------------------
#!/bin/bash
# params: <HOST> <TMPDIR>

HOST=$1
TMPDIR=$2

ROOTPORT=26822

TMPFILE=$TMPDIR/tmp_copy.sql

echo "set autocommit off;" > $TMPFILE
echo "open schema tpc;" >> $TMPFILE
echo "drop table LINEITEM2 cascade;" >> $TMPFILE
echo "drop table ORDERS2 cascade;" >> $TMPFILE
echo "drop table HISTORY2 cascade;" >> $TMPFILE
echo "commit;" >> $TMPFILE

$EXAPLUS -h $HOST -p $ROOTPORT -u tpcuser -P tpcuser -f $TMPFILE
rm -f $TMPFILE




# ------------------------------------------------------------------------------------------
# isolation6.sh
# ------------------------------------------------------------------------------------------
#!/bin/bash
# params: <inputfile> <host> <logdir>

INPUTFILE="$1"
HOST="$2"
LOGDIR="$3"

ROOTPORT=26822

OKEY=( `cat $INPUTFILE` )
SCRIPTSDIR="$HOME/TPCH/ACID/scripts"
LOGFILE=$LOGDIR/isolation6.log

echo "----------------" | tee $LOGFILE
echo "Isolation test 6" | tee -a $LOGFILE
echo "----------------" | tee -a $LOGFILE
echo >> $LOGFILE
echo -n "O_KEY, L_KEY, DELTA = " >> $LOGFILE
cat $INPUTFILE >> $LOGFILE
echo >> $LOGFILE

# Step 1: TPC-H query 1 with DELTA=0
echo "Step 1: TPC-H query 1 with DELTA=0" | tee -a $LOGFILE
$SCRIPTSDIR/tpchquery1ForIsolation6.sh $HOST $LOGDIR 0 >> $LOGFILE &
# wait until the query should be started
sleep 2

# Step 2: ACID transaction
echo "Step 2: ACID transaction" | tee -a $LOGFILE
STARTTIME_TA="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g' | sed 's/  \([0-9]:\)/ 0\1/'`"
echo "starttime of ACID transaction: $STARTTIME_TA" >> $LOGFILE
$SCRIPTSDIR/transaction_isolation/acidtransaction        $HOST        $ROOTPORT        COMMIT        NOWAIT        $INPUTFILE
$LOGDIR/tmp/isolation6_return_t1.log  tpcuser  tpcuser  |  sed  -u  's/^/Transaction  2:/g'  |  tee  -a  $LOGFILE  |  grep
"\(ROLLBACK\|ERROR\)"
ENDTIME_TA="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g' | sed 's/  \([0-9]:\)/ 0\1/'`"
echo "endtime of ACID transaction: $ENDTIME_TA" >> $LOGFILE
wait

if [ ! -z "`grep '\(ROLLBACK\|ERROR\)' $LOGFILE | grep 'Transaction 2'`" ]
then
        ERROR="ERROR Transaction 2"
fi

# ensure the acid transaction completed within the execution time of the query
# i.e. starttime_query < start_time_transaction, endtime_transaction < endtime_query
STARTTIME_QU="`grep 'Q1' $LOGDIR/tmp/Q1Delta.log2 | sed 's/[^;]*; \([^;]*\);.*/\1/'`"
ENDTIME_QU="`grep 'Q1' $LOGDIR/tmp/Q1Delta.log2 | sed 's/[^;]*;[^;]*; \([^;]*\);.*/\1/'`"

echo "STARTTIME_TA=$STARTTIME_TA" >> $LOGFILE
echo "ENDTIME_TA=$ENDTIME_TA" >> $LOGFILE
echo "STARTTIME_QU=$STARTTIME_QU" >> $LOGFILE
echo "ENDTIME_QU=$ENDTIME_QU" >> $LOGFILE

# convert times to format "yyyymmddhhmissff3" for string comparison
```

```
START_QU="`echo  $STARTTIME_QU  |  sed  's/\([0-9][0-9]\)\/\([0-9][0-9]\)\/\([0-9][0-9][0-9][0-9]\)/\3\/\1\/\2/'  |  sed  's/[
\t/:.]*//g'`"
END_QU="`echo  $ENDTIME_QU  |  sed  's/\([0-9][0-9]\)\/\([0-9][0-9]\)\/\([0-9][0-9][0-9][0-9]\)/\3\/\1\/\2/'  |  sed  's/[
\t/:.]*//g'`"
START_TA="`echo  $STARTTIME_TA  |  sed  's/\([0-9][0-9]\)\/\([0-9][0-9]\)\/\([0-9][0-9][0-9][0-9]\)/\3\/\1\/\2/'  |  sed  's/[
\t/:.]*//g'`"
END_TA="`echo  $ENDTIME_TA  |  sed  's/\([0-9][0-9]\)\/\([0-9][0-9]\)\/\([0-9][0-9][0-9][0-9]\)/\3\/\1\/\2/'  |  sed  's/[
\t/:.]*//g'`"

echo "START_TA=$START_TA" >> $LOGFILE
echo "END_TA=$END_TA" >> $LOGFILE
echo "START_QU=$START_QU" >> $LOGFILE
echo "END_QU=$END_QU" >> $LOGFILE

if [ $START_TA -le $START_QU ]
then
         ERROR="$ERROR Transaction started before query"
fi

if [ $END_TA -ge $END_QU ]
then
        ERROR="$ERROR Transaction ended after query"
fi

if [ -z "$ERROR" ]
then
        echo "Isolation test 6 passed." | tee -a $LOGFILE
else
        echo "Isolation test 6 failed." | tee -a $LOGFILE
        echo "$ERROR" | tee -a $LOGFILE
fi



# ------------------------------------------------------------------------------------
# tpchquery1ForIsolation6.sh
# ------------------------------------------------------------------------------------
#!/bin/bash
# params: <HOST> <LOGDIR> <DELTA>

HOST="$1"
LOGDIR="$2"
DELTA="$3"

TMPDIR="$LOGDIR/tmp"
SQL="$HOME/TPCH/ACID/sql/Q1Delta.sql"

QUERYSTREAMDIR="$HOME/TPCH/scripts/query_streams/"

ROOTPORT=26822


# transform inputfile...
cp $SQL $TMPDIR/Q1NOReformat.sql
cat $TMPDIR/Q1NOReformat.sql | sed "s/:q/1/g" | sed "s/:s/1/g" | sed "s/:o//g" | sed "s/:e/COMMIT;/g" | sed "s/:1/$DELTA/g" |
sed "s/\r$//g" > $TMPDIR/Q1Delta.sql

# starting query
$QUERYSTREAMDIR/querystreamex  $TMPDIR/Q1Delta.sql  $TMPDIR/Q1Delta.log  $TMPDIR/Q1Delta.log2  1  $HOST  $ROOTPORT  tpc  tpcuser
tpcuser



# ------------------------------------------------------------------------------------
# Q1Delta.sql
# ------------------------------------------------------------------------------------
-- $ID$
-- TPC-H/TPC-R Pricing Summary Report Query (Q1)
-- Functional Query Definition
-- Approved February 1998

-- Minor modification - date arithmetic for days ( 2.2.3.3 c) )

-- TPC-H Query :q :s
:o

select
        l1.l_quantity,
        sum(l2.l_extendedprice),
        sum(l3.l_extendedprice),
        sum(l3.l_quantity)
from
        lineitem l1, lineitem l2, lineitem l3, lineitem l4
where
        l1.l_shipdate <= date '1998-12-01' - 0
        and l1.l_orderkey = l2.l_orderkey
        and l1.l_linenumber = l2.l_linenumber
        and l1.l_extendedprice = l3.l_extendedprice
        and l3.l_quantity<30
          and l4.l_quantity = l1.l_quantity
          and l4.l_orderkey < 50
group by l1.l_quantity
;
:e



# ------------------------------------------------------------------------------------
```

```
# transaction_isolation/querystream.h
# ----------------------------------------------------------------------------------------
#ifndef QUERYSTREAM_H
#define QUERYSTREAM_H

#include <string>
#include <iostream>

#include "exaCInterface.h"

/** \brief The maximum length of a column name. */
static const size_t maxColumnNameSize = 40;

/** \brief A structure to describe connection information to a database.
  * Used in QueryStream::connectToServer().
  * \sa QueryStream::connectToServer() */
struct Configuration
{
    /** \brief The hostname of the server. */
    char *host;
    /** \brief The port of the server to connect to. */
    char *port;
    /** \brief The user name used for authentication to the database. */
    char *user;
    /** \brief The password of the user. */
    char *password;
};

/** \brief This exception is thrown when a transaction was automatically rolled back.*/
class rollbackException
{
};

/** \brief A structure used to fetch data from a database.
  * Each database column is represented by one FetchStruct.
  * Only one of the arrays of ODBC-C-datatypes is initialized and filled per column.
  * That one is identified by the c_datatype. */
struct FetchStruct
{
    /** \brief Array for columns of SQLCHAR* type
     *(one large block, easy access by sqlCharArrayAccess). */
    SQLCHAR         *sqlCharArray;
    /** \brief To quickly access the i-th string in the sqlCharArray array. */
        SQLCHAR*          *sqlCharArrayAccess;
    /** \brief Array for columns of SQLSMALLINT type. */
    SQLSMALLINT     *sqlSmallInt;
    /** \brief Array for columns of SQLUSMALLINT type. */
    SQLUSMALLINT    *sqlUSmallInt;
    /** \brief Array for columns of SQLINTEGER type. */
    SQLINTEGER      *sqlInteger;
    /** \brief Array for columns of SQLUINTEGER type. */
    SQLUINTEGER     *sqlUInteger;
    /** \brief Array for columns of SQLBIGINT type. */
    SQLBIGINT       *sqlBigInt;
    /** \brief Array for columns of SQLUBIGINT type. */
    SQLUBIGINT      *sqlUBigInt;
    /** \brief Array for columns of SQLREAL type. */
    SQLREAL         *sqlReal;
    /** \brief Array for columns of SQLDOUBLE type. */
    SQLDOUBLE       *sqlDouble;
    /** \brief Array for columns of SQLCHAR type. */
    SQLCHAR         *sqlChar;
    /** \brief Array for columns of SQLSCHAR type. */
    SQLSCHAR        *sqlSChar;
    /** \brief Array for columns of SQL_DATE_STRUCT type. */
    SQL_DATE_STRUCT *sqlDate;
    /** \brief Array for columns of SQL_NUMERIC_STRUCT type. */
    SQL_NUMERIC_STRUCT *sqlNumeric;

    /** \brief The length of a returned data and an indicator whether it is NULL. */
    SQLLEN *resultLen;

    /** \brief The name of the column. */
    SQLCHAR columnName[maxColumnNameSize];
    /** \brief The length of the column name. */
    SQLSMALLINT nameLength;
    /** \brief The SQL datatype ID of the column. */
    SQLSMALLINT datatype;
    /** \brief The ODBC C datatype ID of the column.
      * Might be set by QueryStream::find_ODBC_C_Type( SQLSMALLINT sqlType ) */
    SQLSMALLINT c_datatype;
    /** \brief The size of the column. */
    SQLUINTEGER columnSize;
    /** \brief The number of decimal digits of the column. */
    SQLSMALLINT decimalDigits;
    /** \brief Whether the column allows NULL values. */
    SQLSMALLINT nullable;
};

/** \brief The class represents a query stream to a server.
  * This query stream is slightly different from that one used by the Performance test
  * since it is for implementing the ACID-Transaction for which queries are only
  * executed to get (or change) single lines whose values must be accessible since
  * they will be changed.
  *
  * The execute(...) function was slightly changed to fetch at most two rows and not output or delete
  * the result set. Furthermore a new function freeFetchArray() was introduced to allow the user of
  * this class to free the fetchMe array. Last but not least the fetchMe array of fetched rows is
  * publicly accessable by several getDatatype(colum) functions to allow reading of returned data.
  *
  * In addition to that the mapping of C-Datatypes to SQL-Types might have changed slightly to ensure
```

```
  * mapping of some columns to numerical C-Types. This affects find_ODBC_C_Type(...). */
class QueryStream
{
public:
    /** \brief Constructor.
      * \param streamId The identifier used to identify this query stream.
      * \param logfile The stream to be used for logging. */
    QueryStream( std::string streamId, std::ostream* logfile = &std::cout );
    /** \brief Tries to establish a connection to a server.
      * \param configuration The connection and login information for the server.
      * \return true if the connection could be established, false otherwise. */
    bool connectToServer( Configuration configuration );
    /** \brief Executes a given statement or query.
      * \param statement The text of the statement or query to execute.
      * \param fetchSize The number of rows to fetch with a single fetch command
      * (bigger fetch size might reduce overhead, but increases memory usage).
      * \return The number of rows in resultset (-2 (error), -1 (no resultset), 0,1 or 2 (means more than one)).
      */
    int execute( std::string statement );
    /** \brief Closes the connection to the server.
      * \return true if the connection was closed successfully, false if errors occured. */
    bool disconnectFromServer();
          /** \brief frees the memory allocated to store the resultset.
            * Call this exactly if execute did not return -1. */
          void freeFetchArray();

          int getInt( size_t columnNumber );

          double getDouble( size_t columnNumber );
protected:
    /** \brief The maximum size of a string. */
    static const size_t maxStringSize  = 200;
    /** \brief The number of rows to fetch in a single fetch command. */
    SQLINTEGER fetchBlockSize;
          /** \brief The number to use for fetchBlockSize if it is not
      * given as argument to execute(). */
          static const size_t standardFetchBlockSize = 100;

    /** \brief Binds a column to an entry in the fetchMe array.
      * This must be done before fetching a resultset.
      * This function arranges memory allocation and binding function.
      * \param colNo The number of the resultset column to bind to.
      * \param localColNo The index in the fetchMe array the resultset data shall be stored.
      * \param datatype The SQL datatype of the column to bind to. */
    void bindColumn( int colNo, int localColNo, SQLSMALLINT datatype );

    /** \brief Allocates memory for the buffers needed in binding resultset columns.
      * The array corresponding to the given datatype of the given index
      * of the fetchMe array is initialized to the size of fetchBlockSize.
      * If the datatype is a string datatype, the string buffer will have the
      * size maxStringSize.
      * \param localColNo The index in the fetchMe array the data shall be allocated for.
      * \param datatype The ODBC-C-datatype memory shall be allocated for. */
    void allocateMemoryForBind( int localColNo, SQLSMALLINT datatype );

    /** \brief Maps an SQL-datatype-ID to a ODBC-C-datatype-ID to which it can be converted.
      * This is needed for binding columns and fetching results from a remote database.
      * \param sqlType The SQL-datatype-ID whose corresonding ODBC-C-datatype-ID is wanted.
      * \return The ODBC-C-datatype-ID for the given SQL-datatype-ID */
    SQLSMALLINT find_ODBC_C_Type( SQLSMALLINT sqlType );

    /** \brief Format resultset header for output. */
    void outputHeader();

    /** \brief Formats fetched rows for output.
      * The data in fetchMe is printed for each column and all fetched rows. */
    void outputFetchedRows();

    /** \brief The stream identifier. */
    std::string myID;

    /** \brief The environment handle used. */
    SQLHENV  henv;
    /** \brief The connection handle used. */
    SQLHDBC  hdbc;
    /** \brief The only statement handle used. */
    SQLHSTMT hstmt;

    /** \brief The structure used to fetch result sets of queries. */
    FetchStruct *fetchMe;

    /** \brief The number of rows in a result set. */
    SQLSMALLINT numResultCols;

    /** \brief The number of rows fetched by last fetch command. */
    SQLUINTEGER rowsFetched;

    /** \brief The stream used for logging results.*/
    std::ostream* logfile;

private:
          /** \brief Parses the return code of an statement and gives additional information on error. */
          SQLRETURN execCritical(SQLRETURN retcode, SQLSMALLINT handle_type = 0, SQLHANDLE handle = 0);
};

#endif // QUERYSTREAM_H



# ------------------------------------------------------------------------------
# transaction_isolation/querystream.cpp
```

```
# ------------------------------------------------------------------------------------
#include "querystream.h"
#include <iostream>

//#define DEBUG
#ifdef DEBUG
#define DEBUGOUT( X ) cout << "DEBUG:" << X
#else
#define DEBUGOUT( X )
#endif

#define LOG *logfile

using std::cout;
using std::endl;
using std::cerr;

#include <string>
using std::string;

SQLRETURN QueryStream::execCritical(SQLRETURN retcode, SQLSMALLINT handle_type, SQLHANDLE handle)
{
    switch(retcode)
    {
    case SQL_INVALID_HANDLE:
                cout << "invalid handle" << endl;
        throw "ERROR use of invalid handle!\n";
        break;
    case SQL_ERROR:
          {
                SQLCHAR err[1000];
                SQLCHAR state[6];
                SQLINTEGER native;
                SQLSMALLINT len;
                SQLSMALLINT i=1;
        while ( handle !=0 && EXAGetDiagRec(handle_type, handle, i, state, &native,err,1000,&len) == SQL_SUCCESS )
                {
                        cout << state << ":" << err << endl;
                        i++;
                        if ( string( (const char*) err).find( "GlobalTransactionRollback") )
                        {
                                throw rollbackException();
                        }
                }
                cout << "SQL error" << endl;
        throw "An error occured while executing command!\n";
        break;
          }
          case SQL_SUCCESS_WITH_INFO:
                cout << "Problem: Has Info, which should not be" << endl;
                throw "Problem: Has Info, which should not be!\n";
          break;
          case SQL_STILL_EXECUTING:
                cout << "Still executing. This should not be should only";
                cout << "return on successful execution!" << endl;
                throw "Unexpectedly still executing!\n";
                break;
          case SQL_NO_DATA:
          case SQL_SUCCESS:
        break;
          default:
                // should not reach here...
                cout << "unexpected return code: " << retcode << endl;
                throw "unexpected return code!\n";
                break;
    }
    return retcode;
}

QueryStream::QueryStream( std::string streamId, std::ostream* logfile_ )
{
    myID  = streamId;
    henv  = NULL;
    hdbc  = NULL;
    hstmt = NULL;
    logfile = logfile_;
}

bool QueryStream::connectToServer( Configuration cfg )
{
    char localeAttr[]="deu";
    try
    {
        // Allocating environment handle
        execCritical(EXAAllocHandle(SQL_HANDLE_ENV, NULL, &henv));
        // Setting up environment to german localization
        execCritical(EXASetEnvAttr(henv, EXA_ENV_ATTR_SET_LOCALE_ALL, localeAttr, SQL_NTS));
        // Allocating handle for connection
        execCritical(EXAAllocHandle(SQL_HANDLE_DBC, henv, &hdbc));
        // Turning autocommit off... is this necessary?
        execCritical(EXASetConnectAttr(hdbc, SQL_ATTR_AUTOCOMMIT, (void*)SQL_AUTOCOMMIT_OFF, SQL_NTS) );
        // Setting up clientname
        execCritical(EXASetConnectAttr(hdbc, EXA_CLIENT_NAME, (void*)myID.c_str(), SQL_NTS));
        // Connect to server
        execCritical(EXAServerConnect(hdbc, cfg.host, SQL_NTS, cfg.port, SQL_NTS, cfg.user, SQL_NTS, cfg.password, SQL_NTS));
    }
    catch(...)
    {
        // might be nice to clean up handles
        return false;
    }
```

```
    return true;
}

int QueryStream::execute( std::string statement )
{
    // the number of rows in the resultSet
    int resultSetSize = -1;

    try
    {
        // put the statement in the logfile
        LOG << statement << endl;
        // Allocating statement handle
        execCritical(EXAAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt), SQL_HANDLE_DBC, hdbc);
        SQLCHAR* statementtext = (SQLCHAR*) statement.c_str();
        // execute the statement / query
        execCritical(EXAExecDirect(hstmt, statementtext, statement.size() ), SQL_HANDLE_STMT, hstmt );
        // get the affected rows (only insert, update, delete)
        SQLINTEGER affectedrows;
        execCritical(EXARowCount(hstmt,&affectedrows), SQL_HANDLE_STMT, hstmt);
        // get the number of columns in the resultset (if any)
        execCritical(EXANumResultCols(hstmt,&numResultCols), SQL_HANDLE_STMT, hstmt);
        // no result columns means no query
        if ( numResultCols <= 0 )
            LOG << "affected rows: " << affectedrows << "\n\n";

        // if a resultset was constructed, retrieve and print it
        if ( numResultCols > 0 )
        // a resultset was constructed
        {
                            resultSetSize = 0;
            LOG << '\n';
            // setting number of rows to fetch with each fetch
            SQLUINTEGER fetchsize = 2; // This is to store the result set and be able to ensure only one line was received
                            fetchBlockSize = 2;
            execCritical(EXASetStmtAttr(hstmt, SQL_ATTR_ROW_ARRAY_SIZE,
                                        (SQLPOINTER) fetchsize , 0 ), SQL_HANDLE_STMT, hstmt);
            //cout << "Fetch block size (SQL_ATTR_ROW_ARRAY_SIZE) set to " << fetchsize << endl;
            // setting position to store the fetch size for a query
            execCritical(EXASetStmtAttr(hstmt, SQL_ATTR_ROWS_FETCHED_PTR,
                                        (SQLPOINTER) &rowsFetched, 0), SQL_HANDLE_STMT, hstmt);
            // allocate the fetch struct to be large enough for the columns
            fetchMe = new FetchStruct[numResultCols];
            // set all pointers to 0 to make it easy to free the correct memory later
            for ( int i=0; i<numResultCols; i++ )
            {
                fetchMe[i].sqlCharArray = 0;
                                    fetchMe[i].sqlCharArrayAccess = 0;
                fetchMe[i].sqlSmallInt  = 0;
                fetchMe[i].sqlUSmallInt = 0;
                fetchMe[i].sqlInteger   = 0;
                fetchMe[i].sqlUInteger  = 0;
                fetchMe[i].sqlBigInt    = 0;
                fetchMe[i].sqlUBigInt   = 0;
                fetchMe[i].sqlReal      = 0;
                fetchMe[i].sqlDouble    = 0;
                fetchMe[i].sqlChar      = 0;
                fetchMe[i].sqlSChar     = 0;
                fetchMe[i].sqlDate      = 0;
                fetchMe[i].sqlNumeric   = 0;
                fetchMe[i].resultLen    = 0;
            }
            execCritical(EXASetStmtAttr(hstmt, SQL_ATTR_ROWS_FETCHED_PTR,
                                    (SQLPOINTER) &rowsFetched, 0), SQL_HANDLE_STMT, hstmt);
            // determine the datatype of the column and additional information.
            for ( SQLSMALLINT colNr=0; colNr<numResultCols; colNr++ )
            {
                // Get column description for the column with number colNr
                EXADescribeCol( hstmt, colNr+1, fetchMe[colNr].columnName,
                                maxColumnNameSize, &(fetchMe[colNr].nameLength),
                                &(fetchMe[colNr].datatype), &(fetchMe[colNr].columnSize),
                                &(fetchMe[colNr].decimalDigits), &(fetchMe[colNr].nullable) );
                fetchMe[colNr].c_datatype = find_ODBC_C_Type( fetchMe[colNr].datatype );

                // bind the column to an entry in the fetchMe array
                bindColumn( colNr+1, colNr, fetchMe[colNr].c_datatype );
            }
            // output the resultset header
            outputHeader();
            // fetch the columns
            do
            {
                // fetch columns
                execCritical(EXAFetch( hstmt ), SQL_HANDLE_STMT, hstmt);
                // output columns
                outputFetchedRows();
                // correct the result set size
                resultSetSize += rowsFetched;
            } while ( rowsFetched == fetchBlockSize );
            // add the total resultset size to the logfile
            LOG << "\nresultset consists of " << resultSetSize << " rows in " << numResultCols << " columns\n\n";
        }
    }
          catch(rollbackException)
          {
        // free the handle for the statement again
        cout << "\nROLLBACK EXECUTING STATEMENT:\n" << statement << '\n' << endl;
        execCritical(EXAFreeHandle( SQL_HANDLE_STMT, hstmt ), SQL_HANDLE_STMT, hstmt);
                return -3;
          }
          catch( char * c )
```

```
            {
        LOG << "ERROR EXECUTING STATEMENT\n\n";
        cout << "\nERROR EXECUTING STATEMENT:\n" << statement << '\n' << endl;
                cout << c << endl;
        execCritical(EXAFreeHandle( SQL_HANDLE_STMT, hstmt ), SQL_HANDLE_STMT, hstmt);
                return -2;
            }
    catch(...)
    {
        LOG << "ERROR EXECUTING STATEMENT\n\n";
        cout << "\nERROR EXECUTING STATEMENT:\n" << statement << '\n' << endl;
        execCritical(EXAFreeHandle( SQL_HANDLE_STMT, hstmt ), SQL_HANDLE_STMT, hstmt);
                return -2;
    }
    execCritical(EXAFreeHandle( SQL_HANDLE_STMT, hstmt ), SQL_HANDLE_STMT, hstmt);
          return resultSetSize;
}

void QueryStream::freeFetchArray()
{
          for ( int i=0; i<numResultCols; i++ )
    {
        if ( fetchMe[i].sqlCharArray != 0 )
        {
            delete[] fetchMe[i].sqlCharArray;
            delete[] fetchMe[i].sqlCharArrayAccess;
        }
        if ( fetchMe[i].sqlSmallInt  != 0 ) delete[] fetchMe[i].sqlSmallInt;
        if ( fetchMe[i].sqlUSmallInt != 0 ) delete[] fetchMe[i].sqlUSmallInt;
        if ( fetchMe[i].sqlInteger   != 0 ) delete[] fetchMe[i].sqlInteger;
        if ( fetchMe[i].sqlUInteger  != 0 ) delete[] fetchMe[i].sqlUInteger;
        if ( fetchMe[i].sqlBigInt    != 0 ) delete[] fetchMe[i].sqlBigInt;
        if ( fetchMe[i].sqlUBigInt   != 0 ) delete[] fetchMe[i].sqlUBigInt;
        if ( fetchMe[i].sqlReal      != 0 ) delete[] fetchMe[i].sqlReal;
        if ( fetchMe[i].sqlDouble    != 0 ) delete[] fetchMe[i].sqlDouble;
        if ( fetchMe[i].sqlChar      != 0 ) delete[] fetchMe[i].sqlChar;
        if ( fetchMe[i].sqlSChar     != 0 ) delete[] fetchMe[i].sqlSChar;
        if ( fetchMe[i].sqlDate      != 0 ) delete[] fetchMe[i].sqlDate;
        if ( fetchMe[i].sqlNumeric   != 0 ) delete[] fetchMe[i].sqlNumeric;
        if ( fetchMe[i].resultLen    != 0 ) delete[] fetchMe[i].resultLen;
    }
    delete[] fetchMe;
}


int QueryStream::getInt( size_t columnNumber )
{
          size_t i = columnNumber;
          if ( i >= numResultCols )
          {
                    cout << "column out of range" << endl;
                    throw "column out of range";
          }

          if ( fetchMe[i].sqlCharArray != 0 ) return atoi( (char*) fetchMe[i].sqlCharArrayAccess[0] );

          if ( fetchMe[i].sqlSmallInt  != 0 ) return fetchMe[i].sqlSmallInt[0];
    if ( fetchMe[i].sqlUSmallInt != 0 ) return fetchMe[i].sqlUSmallInt[0];
    if ( fetchMe[i].sqlInteger   != 0 ) return fetchMe[i].sqlInteger[0];
    if ( fetchMe[i].sqlUInteger  != 0 ) return fetchMe[i].sqlUInteger[0];
    if ( fetchMe[i].sqlBigInt    != 0 ) return fetchMe[i].sqlBigInt[0];
    if ( fetchMe[i].sqlUBigInt   != 0 ) return fetchMe[i].sqlUBigInt[0];
    if ( fetchMe[i].sqlReal      != 0 ) return (int) fetchMe[i].sqlReal[0];
    if ( fetchMe[i].sqlDouble    != 0 ) return (int) fetchMe[i].sqlDouble[0];
    if ( fetchMe[i].sqlChar      != 0 ) return fetchMe[i].sqlChar[0];
    if ( fetchMe[i].sqlSChar     != 0 ) return fetchMe[i].sqlSChar[0];

          cout << "cannot cast datatype to Int" << endl;
          throw "cannot cast datatype to Int";
}

double QueryStream::getDouble( size_t columnNumber )
{
          size_t i = columnNumber;
          if ( i >= numResultCols )
          {
                    cout << "column out of range" << endl;
                    throw "column out of range";
          }

          if ( fetchMe[i].sqlCharArray != 0 ) return atof( (char*) fetchMe[i].sqlCharArrayAccess[0] );

          if ( fetchMe[i].sqlSmallInt  != 0 ) return fetchMe[i].sqlSmallInt[0];
    if ( fetchMe[i].sqlUSmallInt != 0 ) return fetchMe[i].sqlUSmallInt[0];
    if ( fetchMe[i].sqlInteger   != 0 ) return fetchMe[i].sqlInteger[0];
    if ( fetchMe[i].sqlUInteger  != 0 ) return fetchMe[i].sqlUInteger[0];
    if ( fetchMe[i].sqlBigInt    != 0 ) return fetchMe[i].sqlBigInt[0];
    if ( fetchMe[i].sqlUBigInt   != 0 ) return fetchMe[i].sqlUBigInt[0];
    if ( fetchMe[i].sqlReal      != 0 ) return fetchMe[i].sqlReal[0];
    if ( fetchMe[i].sqlDouble    != 0 ) return fetchMe[i].sqlDouble[0];
    if ( fetchMe[i].sqlChar      != 0 ) return fetchMe[i].sqlChar[0];
    if ( fetchMe[i].sqlSChar     != 0 ) return fetchMe[i].sqlSChar[0];

          cout << "cannot cast datatype to Int" << endl;
          throw "cannot cast datatype to Int";

}

bool QueryStream::disconnectFromServer()
{
    try
```

```
    {
        // disconnecting from server
        execCritical(EXADisconnect(hdbc));
        // freeing connection handle
        execCritical(EXAFreeHandle(SQL_HANDLE_DBC, hdbc));
        // freeing environment handle
        execCritical(EXAFreeHandle(SQL_HANDLE_ENV, henv));
    }
    catch(...)
    {
        cout << "Error while disconnecting from server" << endl;
        return false;
    }
    return true;
}

void QueryStream::outputHeader()
{
    int headerLength = 1;
    LOG << "|";
    for ( size_t col=0; col<numResultCols; col++ )
    {
        LOG << fetchMe[col].columnName << "|";
        headerLength += fetchMe[col].nameLength + 1;
    }
    LOG << '\n';
    for ( int i=0; i<headerLength; i++ )
        LOG << '=';
    LOG << '\n';
          (LOG).flush();
}

void QueryStream::outputFetchedRows()
{
    for ( size_t rows=0; rows<rowsFetched; rows++ )
    {
        LOG << "|";
        for ( size_t cols=0; cols<numResultCols; cols++ )
        {
            if ( fetchMe[cols].resultLen[rows] == SQL_NULL_DATA )
            {
                LOG << "|";
                continue;
            }
            switch( fetchMe[cols].c_datatype )
            {
                // string data
                case SQL_C_CHAR:
                case SQL_C_BINARY:
                //case SQL_C_XML:
                //case SQL_C_VARBOOKMARK:
                    LOG << fetchMe[cols].sqlCharArrayAccess[rows] << "|";
                break;

                // small int
                case SQL_C_SSHORT:
                    LOG << fetchMe[cols].sqlSmallInt[rows] << "|";
                break;

                // unsigned small int
                case SQL_C_USHORT:
                    LOG << fetchMe[cols].sqlUSmallInt[rows] << "|";
                break;

                // long int
                case SQL_C_SLONG:
                    LOG << fetchMe[cols].sqlInteger[rows] << "|";
                break;

                // unsigned long int
                case SQL_C_ULONG:
                    LOG << fetchMe[cols].sqlUInteger[rows] << "|";
                break;

                // float
                case SQL_C_FLOAT:
                    LOG << fetchMe[cols].sqlReal[rows] << "|";
                break;

                // double
                case SQL_C_DOUBLE:
                    LOG << fetchMe[cols].sqlDouble[rows] << "|";
                break;

                // unsigned char
                case SQL_C_BIT:
                case SQL_C_UTINYINT:
                    LOG << fetchMe[cols].sqlChar[rows] << "|";
                break;

                // signed char
                case SQL_C_STINYINT:
                    LOG << fetchMe[cols].sqlSChar[rows] << "|";
                break;

                // big int
                case SQL_C_SBIGINT:
                    LOG << fetchMe[cols].sqlBigInt[rows] << "|";
                break;

                // unsigned big int
```

```
case SQL_C_UBIGINT:
    LOG << fetchMe[cols].sqlUBigInt[rows] << "|";
break;

// date
case SQL_C_TYPE_DATE:
{
    // format date to YYYY-MM-DD for output
    if ( fetchMe[cols].sqlDate[rows].year < 1000 )
        LOG << "0";
    if ( fetchMe[cols].sqlDate[rows].year < 100 )
        LOG << "0";
    if ( fetchMe[cols].sqlDate[rows].year < 10 )
        LOG << "0";
    LOG << fetchMe[cols].sqlDate[rows].year << "-";

    if ( fetchMe[cols].sqlDate[rows].month < 10 )
        LOG << "0";
    LOG << fetchMe[cols].sqlDate[rows].month << "-";

    if ( fetchMe[cols].sqlDate[rows].day < 10 )
        LOG << "0";
    LOG << fetchMe[cols].sqlDate[rows].day << "|";
}
break;

// numeric, i.e. decimal
                        // not used since it is eaier by far to map decimals to doubles
case SQL_C_NUMERIC:
{
    // format decimal for output
    int scale = fetchMe[cols].sqlNumeric[rows].scale;
    char text[3*SQL_MAX_NUMERIC_LEN+2+scale];
    int pos_T = 3*SQL_MAX_NUMERIC_LEN+scale;
    int pos_N = SQL_MAX_NUMERIC_LEN-1;
    text[3*SQL_MAX_NUMERIC_LEN+1+scale] = '\0';
    //int add = 0;
    //if ( fetchMe[cols].sqlNumeric[rows].sign == 0 )
    //    add=1;
    while ( fetchMe[cols].sqlNumeric[rows].val[pos_N] == 0 && pos_N > 0)
        pos_N--;

    while ( pos_N >= 0 )
    {
        int current = fetchMe[cols].sqlNumeric[rows].val[pos_N];
        int newVal = 0;
        int currentPos = pos_N;
        while ( currentPos >= 0 )
        {
            //if ( currentPos == 0 )
            //{
            //    current += add;
            //    add = 0;
            //}
            if ( current < 10 )
            {

                fetchMe[cols].sqlNumeric[rows].val[currentPos] = newVal;
                currentPos--;

                if ( currentPos >= 0 )
                {
                    current = current * 256 + fetchMe[cols].sqlNumeric[rows].val[currentPos];
                    newVal = 0;
                }
            }
            else
            {
                newVal = current / 10;
                current = current % 10;
            }
        }
        if ( fetchMe[cols].sqlNumeric[rows].val[pos_N] == 0 )
        {
            pos_N--;
        }
        text[pos_T--] = '0' + current;
        if ( scale == 1 )
        {
            text[pos_T--] = '.';
        }
        scale--;
    }
    if ( scale==0)
        text[pos_T--] = '0';
    while ( scale > 0 )
    {
        text[pos_T--] = '0';
        if ( scale == 1 )
        {
            text[pos_T--] = '.';
            text[pos_T--] = '0';
        }
        scale--;
    }

    if ( fetchMe[cols].sqlNumeric[rows].sign == 0 )
        text[pos_T--] = '-';
    LOG << text+pos_T+1 << "|";
}
break;
```

```
                // other datatype
                default:
                    cout << "output: datatype " << fetchMe[cols].c_datatype << " not supported yet" << endl;
                    throw fetchMe[cols].c_datatype;
                break;
            }
        }
        LOG << '\n';
                (LOG).flush();
    }
}

void QueryStream::bindColumn( int colNo, int localColNo, SQLSMALLINT c_datatype )
{
    // allocate memory for the column to bind
    allocateMemoryForBind( localColNo, c_datatype );
    // bind the column
    switch ( c_datatype )
    {
        // string data
        case SQL_C_CHAR:
        case SQL_C_BINARY:
        //case SQL_C_XML:
        //case SQL_C_VARBOOKMARK:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlCharArray,
                    maxStringSize*sizeof(SQLCHAR), fetchMe[localColNo].resultLen));
        break;

        // small int
        case SQL_C_SSHORT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlSmallInt,
                    sizeof(SQLSMALLINT), fetchMe[localColNo].resultLen));
        break;

        // unsigned small int
        case SQL_C_USHORT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlUSmallInt,
                    sizeof(SQLUSMALLINT), fetchMe[localColNo].resultLen));
        break;

        // long int
        case SQL_C_SLONG:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlInteger,
                    sizeof(SQLINTEGER), fetchMe[localColNo].resultLen));
        break;

        // unsigned long int
        case SQL_C_ULONG:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlUInteger,
                    sizeof(SQLUINTEGER), fetchMe[localColNo].resultLen));
        break;

        // float
        case SQL_C_FLOAT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlReal,
                    sizeof(SQLREAL), fetchMe[localColNo].resultLen));
        break;

        // double
        case SQL_C_DOUBLE:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlDouble,
                    sizeof(SQLDOUBLE), fetchMe[localColNo].resultLen));
        break;

        // unsigned char
        case SQL_C_BIT:
        case SQL_C_UTINYINT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlChar,
                    sizeof(SQLCHAR), fetchMe[localColNo].resultLen));
        break;

        // signed char
        case SQL_C_STINYINT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlSmallInt,
                    sizeof(SQLSCHAR), fetchMe[localColNo].resultLen));
        break;

        // big int
        case SQL_C_SBIGINT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlBigInt,
                    sizeof(SQLBIGINT), fetchMe[localColNo].resultLen));
        break;

        // unsigned big int
        case SQL_C_UBIGINT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlUBigInt,
                    sizeof(SQLUBIGINT), fetchMe[localColNo].resultLen));
        break;
```

```
        // date
        case SQL_C_TYPE_DATE:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlDate,
                    sizeof(SQL_DATE_STRUCT), fetchMe[localColNo].resultLen));
        break;

        // numeric, i.e. decimal
        case SQL_C_NUMERIC:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlNumeric,
                    sizeof(SQL_NUMERIC_STRUCT), fetchMe[localColNo].resultLen));
        break;

        // other datatype
        default:
            throw c_datatype;
        break;
    }
}

void QueryStream::allocateMemoryForBind( int localColNo, SQLSMALLINT datatype )
{
    // allocate memory for resultLen field used to indicate NULL values (among others)
    fetchMe[localColNo].resultLen = new SQLLEN[fetchBlockSize];

    // allocate memory depending on the column datatype
    switch ( datatype )
    {
        // string data
        case SQL_C_CHAR:
        case SQL_C_BINARY:
        //case SQL_C_XML:
        //case SQL_C_VARBOOKMARK:
            fetchMe[localColNo].sqlCharArray = new SQLCHAR[fetchBlockSize*maxStringSize];
                        fetchMe[localColNo].sqlCharArrayAccess = new SQLCHAR*[fetchBlockSize];
            for ( size_t i=0; i<fetchBlockSize; i++ )
            {
                fetchMe[localColNo].sqlCharArrayAccess[i] = fetchMe[localColNo].sqlCharArray+i*maxStringSize;
            }
        break;

        // small int
        case SQL_C_SSHORT:
            fetchMe[localColNo].sqlSmallInt = new SQLSMALLINT[fetchBlockSize];
        break;

        // unsigned small int
        case SQL_C_USHORT:
            fetchMe[localColNo].sqlUSmallInt = new SQLUSMALLINT[fetchBlockSize];
        break;

        // long int
        case SQL_C_SLONG:
            fetchMe[localColNo].sqlInteger = new SQLINTEGER[fetchBlockSize];
        break;

        // unsigned long int
        case SQL_C_ULONG:
            fetchMe[localColNo].sqlUInteger = new SQLUINTEGER[fetchBlockSize];
        break;

        // float
        case SQL_C_FLOAT:
            fetchMe[localColNo].sqlReal = new SQLREAL[fetchBlockSize];
        break;

        // double
        case SQL_C_DOUBLE:
            fetchMe[localColNo].sqlDouble = new SQLDOUBLE[fetchBlockSize];
        break;

        // unsigned char
        case SQL_C_BIT:
        case SQL_C_UTINYINT:
            fetchMe[localColNo].sqlChar = new SQLCHAR[fetchBlockSize];
        break;

        // signed char
        case SQL_C_STINYINT:
            fetchMe[localColNo].sqlSChar = new SQLSCHAR[fetchBlockSize];
        break;

        // big int
        case SQL_C_SBIGINT:
            fetchMe[localColNo].sqlBigInt = new SQLBIGINT[fetchBlockSize];
        break;

        // unsigned big int
        case SQL_C_UBIGINT:
            fetchMe[localColNo].sqlUBigInt = new SQLUBIGINT[fetchBlockSize];
        break;

        // date
        case SQL_C_TYPE_DATE:
            fetchMe[localColNo].sqlDate = new SQL_DATE_STRUCT[fetchBlockSize];
        break;

        // numeric, i.e. decimal
        case SQL_C_NUMERIC:
            fetchMe[localColNo].sqlNumeric = new SQL_NUMERIC_STRUCT[fetchBlockSize];
```

```
        break;

        // other datatype
        default:
            cout << "memory allocation: datatype " << datatype << " not supported yet" << endl;
            throw datatype;
        break;
    }
}

SQLSMALLINT QueryStream::find_ODBC_C_Type( SQLSMALLINT sqlType )
{
    switch( sqlType )
    {
        case SQL_CHAR:
        case SQL_VARCHAR:
        case SQL_LONGVARCHAR:
        case SQL_WCHAR:
        case SQL_WVARCHAR:
        case SQL_WLONGVARCHAR:
            return SQL_C_CHAR;
        break;

        case SQL_DECIMAL:
        case SQL_NUMERIC:
            //return SQL_C_NUMERIC;
            return SQL_C_DOUBLE;
            //return SQL_C_CHAR;
        break;

        case SQL_SMALLINT:
            return SQL_C_SSHORT;
        break;

        case SQL_INTEGER:
            return SQL_C_SLONG;
        break;

        case SQL_REAL:
        case SQL_FLOAT:
        case SQL_DOUBLE:
            return SQL_C_DOUBLE;
        break;

        case SQL_BIT:
            return SQL_C_BIT;
        break;

        case SQL_TINYINT:
            return SQL_C_STINYINT;
        break;

        case SQL_BIGINT:
            return SQL_C_SBIGINT;
        break;

        case SQL_BINARY:
        case SQL_VARBINARY:
        case SQL_LONGVARBINARY:
            return SQL_C_BINARY;
        break;

        case SQL_TYPE_DATE:
            return SQL_C_TYPE_DATE;
        break;

        default:
            cout << "unsupported datatype, don't know corresponding C type: " << sqlType << endl;
            throw sqlType;
        break;
    }
}
```

```
# ------------------------------------------------------------------------------------
# run_durability_test.sh (consistency check included)
# ------------------------------------------------------------------------------------
#!/bin/bash
# params: <host> <scalefactor> <streams> <logdir> <number_of_durability_tests>

HOST="$1"
SF="$2"
STREAMS="$3"
LOGDIR="$4/durability"
NUM_TESTS="$5"
mkdir -p $LOGDIR
mkdir -p $LOGDIR/tmp
LOGFILE=$LOGDIR/durability.log
ACIDDIR="$HOME/TPCH/ACID"
TRANSACTIONS_FOR_DURABILITY=400

ROOTPORT=26822

echo "***************" | tee $LOGFILE
echo "DURABILITY TEST" | tee -a $LOGFILE
echo "***************" | tee -a $LOGFILE

## create table history2 (needed for easier comparison of history table and success files)
```

```
echo "creating table history2 (needed for easier comparison of history table and success files)" | tee -a $LOGFILE
$EXAPLUS -h $HOST -p $ROOTPORT -u tpcuser -P tpcuser -f $ACIDDIR/sql/create_history_table2.sql >> /dev/null

## start one durability test after another
for (( ID=1;ID<=$NUM_TESTS; ID++ ))
do
        $ACIDDIR/scripts/durability_test_main.sh  $HOST  $SF  $STREAMS  $LOGDIR  $TRANSACTIONS_FOR_DURABILITY  $ID  |  tee  -a
$LOGFILE
done

## check whether whole durability tests has passed or failed
if [ -z "`grep 'failed' $LOGFILE`" ]
then
        echo "Durability test passed." | tee -a $LOGFILE
else
        echo "Durability test failed. See $LOGDIR/durability.log for details." | tee -a $LOGFILE
fi
```

```
# ------------------------------------------------------------------------------------------
# durability_test_main.sh
# ------------------------------------------------------------------------------------------
#!/bin/bash
# executes a durability test
# param: <roothost> <scalefactor> <streams> <logdir> <numberOfTransactionsPerStream> <testID>

ROOTHOST="$1"
SCALEFACTOR="$2"
STREAMS="$3"
LOGDIR="$4"
NUMTRANSACTIONS="$5"
ID="$6"
LOGFILE="$LOGDIR/durability$ID.log"

ROOTPORT=26822

ACIDDIR="$HOME/TPCH/ACID"
INFORM_AFTER_NUM_TRANSACTIONS=100

export ADDITIONALWAITTIME=5
# unset ADDITIONALWAITTIME

export WAITAFTERROLLBACK=8
# unset WAITAFTERROLLBACK

export WAITAFTERCOMMIT=15
# unset WAITAFTERCOMMIT

# export WAITBEFORECOMMIT=1
unset WAITBEFORECOMMIT

echo "-----------------" | tee $LOGFILE
echo "Durability test $ID" | tee -a $LOGFILE
echo "-----------------" | tee -a $LOGFILE

echo "deleting old entries in history table" | tee -a $LOGFILE
## delete all entries in history table
$EXAPLUS -h $ROOTHOST -p $ROOTPORT -u tpcuser -P tpcuser -f $ACIDDIR/sql/clear_history.sql >> $LOGFILE

## generate random orderkeys for use in consistency check
for (( i=0; i<=$STREAMS; i++ ))
do
        echo "creating $NUMTRANSACTIONS random transactions for stream $i" | tee -a $LOGFILE
        $ACIDDIR/scripts/create_random_transactions.sh \
                        $i $NUMTRANSACTIONS $SCALEFACTOR $ROOTHOST \
                        $LOGDIR/tmp $LOGDIR/durability${ID}_keys$i.log >> $LOGFILE
done

## check initial consistency
INCONSISTENT=(`$ACIDDIR/scripts/check_consistency.sh $ROOTHOST $LOGDIR/durability${ID}_initial_consistency.log`)

if [ $INCONSISTENT -gt 0 ]
then
        echo "Durability Test $ID failed ( Initial consistency check failed, $INCONSISTENT Orderkeys inconsistent )." | tee
-a $LOGFILE
        exit
fi

## start the transactions on all streams
echo "starting stream 0" | tee -a $LOGFILE
$ACIDDIR/scripts/transaction/acidtransaction    $ROOTHOST    $ROOTPORT    COMMIT    NOWAIT    $LOGDIR/durability${ID}_keys0.log
$LOGDIR/durability${ID}_return0.log   tpcuser   tpcuser   $LOGDIR/durability${ID}_values0.log   $INFORM_AFTER_NUM_TRANSACTIONS
$LOGDIR/durability${ID}_success0.log \
              2>/dev/null | tee $LOGDIR/durability${ID}_transaction_0.log &
for (( i=1; i<=$STREAMS; i++ ))
do
        echo "starting stream $i" | tee -a $LOGFILE
        $ACIDDIR/scripts/transaction/acidtransaction $ROOTHOST $ROOTPORT COMMIT NOWAIT $LOGDIR/durability${ID}_keys$i.log
$LOGDIR/durability${ID}_return$i.log  tpcuser  tpcuser  $LOGDIR/durability${ID}_values$i.log  $INFORM_AFTER_NUM_TRANSACTIONS
$LOGDIR/durability${ID}_success$i.log \
                2>/dev/null  |  tee  $LOGDIR/durability${ID}_transaction_$i.log  |  grep  "\(ERROR\|ROLLBACK\|committed  or
rolled back\)" | tee $LOGDIR/tmp/dur${ID}_t$i.err | grep "\(ERROR\|committed or rolled back\)"  \
                  | sed -u "s/^/STREAM $i:/g" | tee -a $LOGFILE &
done

echo "waiting until $INFORM_AFTER_NUM_TRANSACTIONS transactions have been committed or rolled back by each stream." | tee -a
$LOGFILE
```

```
## wait until failure may be caused
# some loop
DONE="NOT DONE"
while [ ! -z "$DONE" ]
do
        DONE=""
        for (( i=0; i<=$STREAMS; i++ ))
        do
                CURRENT="`grep 'committed or rolled back' $LOGDIR/durability${ID}_transaction_$i.log`"
                if [ -z "$CURRENT" ]
                then
                        DONE="NOT DONE"
                fi
        done
        sleep 10
done

## ask for keypress before failure (to check for errors in streamlogs)
echo "Press enter key, please, before causing an error to allow checking for errors (which cause the test to fail)" | tee -a
$LOGFILE
read $X
# check for errors in all stream logs
for (( i=0; i<=$STREAMS; i++ ))
do
        CURRENT="`grep 'ERROR' $LOGDIR/durability${ID}_transaction_$i.log`"
        if [ ! -z "$CURRENT" ]
        then
                ERROR="$ERROR:Error in stream $i"
        fi
done
echo "`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`: Error checking completed" | tee -a $LOGFILE
## kill the streams (if still running)
echo "Press enter key to kill running transactions" | tee -a $LOGFILE
read $X
echo "`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`: killing all acidtransactions which might still be
running" | tee -a $LOGFILE
killall acidtransaction | tee -a $LOGFILE
echo "`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`: killed all acidtransactions" | tee -a $LOGFILE

## wait for keypress after recovery
echo "Press enter key, please, when recovery is completed." | tee -a $LOGFILE
read $X
echo "`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`: Checking consistency" | tee -a $LOGFILE

## check consistency again
INCONSISTENT=(`$ACIDDIR/scripts/check_consistency.sh $ROOTHOST $LOGDIR/durability${ID}_final_consistency.log`)

if [ $INCONSISTENT -gt 0 ]
then
    echo "Durability Test $ID failed ( Final consistency check failed, $INCONSISTENT Orderkeys inconsistent )."
    exit
fi

## clear second history table (easier comparison)
echo "clearing table history2" | tee -a $LOGFILE
$EXAPLUS -h $ROOTHOST -p $ROOTPORT -u tpcuser -P tpcuser -f $ACIDDIR/sql/clear_history2.sql >> $LOGFILE

## insert all entries from all success files in second history table
echo "inserting entries from all success files in table history2" | tee -a $LOGFILE
SQLFILE=$LOGDIR/durability${ID}_history.sql
echo "set autocommit off;" > $SQLFILE
echo "open schema tpc;" >> $SQLFILE
for (( i=0; i<=$STREAMS; i++ ))
do
        cat $LOGDIR/durability${ID}_success$i.log | sed "s/|/,/g" | grep -v CURRENT_DATE_TIME | sed "s/, \(....-\)/,\'\1/g"
| sed "s/^,/insert into history2 values (/g" | sed "s/ ,$/\');/g" >> $SQLFILE
done
echo "commit;" >> $SQLFILE

# load sql into history2
cat $SQLFILE | $EXAPLUS -h $ROOTHOST -p $ROOTPORT -u tpcuser -P tpcuser >> $LOGDIR/durability${ID}_load_into_history2.log

## compare history tables
echo "comparing history tables" | tee -a $LOGFILE
cat   $ACIDDIR/sql/compare_history1.sql   |   $EXAPLUS   -h   $ROOTHOST   -p   $ROOTPORT   -u   tpcuser   -P   tpcuser   >
$LOGDIR/durability${ID}_compare_history1.log
cat   $ACIDDIR/sql/compare_history2.sql   |   $EXAPLUS   -h   $ROOTHOST   -p   $ROOTPORT   -u   tpcuser   -P   tpcuser   >
$LOGDIR/durability${ID}_compare_history2.log
DIFFERENCES1=(`grep 'rows in resultset\.' $LOGDIR/durability${ID}_compare_history1.log | sed 's/rows in resultset\.//g'`)
DIFFERENCES2=(`grep 'rows in resultset\.' $LOGDIR/durability${ID}_compare_history2.log | sed 's/rows in resultset\.//g'`)

echo "storing history table in $LOGDIR/durability${ID}_store_history.log" | tee -a $LOGFILE
cat   $ACIDDIR/sql/store_history.sql   |   $EXAPLUS   -h   $ROOTHOST   -p   $ROOTPORT   -u   tpcuser   -P   tpcuser   >
$LOGDIR/durability${ID}_store_history.log
cat   $ACIDDIR/sql/store_history2.sql   |   $EXAPLUS   -h   $ROOTHOST   -p   $ROOTPORT   -u   tpcuser   -P   tpcuser   >
$LOGDIR/durability${ID}_store_history2.log

## very probably history tables are identical... but there might be one or two additional entries
## if there are differences in history tables, take a look at them yourself to decide whether test was ok
if [ $DIFFERENCES1 -gt 0 ]
then
        echo "Durability test $ID might have failed. See $LOGDIR/durability${ID}_compare_history1.log for details." | tee -a
$LOGFILE
        echo   "You   might   also   want   to   take   a   look   at   $LOGDIR/durability${ID}_store_history.log   and
$LOGDIR/durability${ID}_store_history2.log." | tee -a $LOGFILE
        exit
fi

if [ $DIFFERENCES2 -gt 0 ]
then
```

```
        echo "Durability test $ID might have failed. See $LOGDIR/durability${ID}_compare_history2.log for details." | tee -a
$LOGFILE
        echo    "You    might    also    want    to    take    a    look    at    $LOGDIR/durability${ID}_store_history.log    and
$LOGDIR/durability${ID}_store_history2.log." | tee -a $LOGFILE
        exit
fi

## passed the durability test if there were no errors until now
if [ -z "$ERROR" ]
then
        echo "Durability Test $ID passed." | tee -a $LOGFILE
else
        echo "Durability Test $ID failed due to errors." | tee -a $LOGFILE
        echo "$ERROR" | tee -a $LOGFILE
fi
```

```
# ------------------------------------------------------------------------------------------
# check_consistency.sh
# ------------------------------------------------------------------------------------------
#!/bin/bash
# param <roothost> <logfile>

ROOTHOST="$1"
LOGFILE="$2"
ROOTPORT=26822

SQLDIR="$HOME/TPCH/ACID/sql"

## execute both created queries
cat  $SQLDIR/check_consistency.sql  |  $EXAPLUS  -tabCompletion  off  -autoCompletion  off  -h  $ROOTHOST  -p  $ROOTPORT  -u  sys  -P
exasol > $LOGFILE
SELECTED=(`cat $LOGFILE | grep "rows in resultset" | sed "s/rows in resultset//g"`)

echo "$SELECTED"
```

```
# ------------------------------------------------------------------------------------------
# check_consistency.sql
# ------------------------------------------------------------------------------------------
set autocommit off;

open schema tpc;

select *
from (
        select o_orderkey, o_totalprice - sum( trunc( trunc ( l_extendedprice * (1-l_discount),2)*(1+l_tax),2)) part_res
        from orders, lineitem
        where o_orderkey=l_orderkey
        group by o_orderkey, o_totalprice
) where not part_res=0;


exit;
```

```
# ------------------------------------------------------------------------------------------
# compare_history1.sql
# ------------------------------------------------------------------------------------------
set autocommit off;
open schema tpc;
select * from history h1 where not exists
(
        select *
        from history2 h2
        where h1.h_p_key=h2.h_p_key
                and h1.h_s_key = h2.h_s_key
                and h1.h_o_key=h2.h_o_key
                and h1.h_l_key=h2.h_l_key
                and h1.h_delta=h2.h_delta
);
commit;
```

```
# ------------------------------------------------------------------------------------------
# compare_history2.sql
# ------------------------------------------------------------------------------------------
set autocommit off;
open schema tpc;
select * from history2 h2 where not exists
(
        select *
        from history h1
        where h1.h_p_key=h2.h_p_key
                and h1.h_s_key = h2.h_s_key
                and h1.h_o_key=h2.h_o_key
                and h1.h_l_key=h2.h_l_key
                and h1.h_delta=h2.h_delta
);
commit;
```

```
# ------------------------------------------------------------------------------------------
# create_random_transactions.sh
# ------------------------------------------------------------------------------------------
#!/bin/bash
# params: <stream> <count> <scalefactor> <host> <tmpdir> <outfile>
STREAM=$1
COUNT=$2
SF=$3
HOST=$4
TMPFILE=$5/transactions_$STREAM.sql
OUTFILE=$6
ROOTPORT=26822

echo "set autocommit off;" > $TMPFILE
echo >> $TMPFILE
echo "open schema tpc;" >> $TMPFILE
echo >> $TMPFILE
echo "create table indices$STREAM ( i dec(10) );" >> $TMPFILE
echo -n "insert into indices$STREAM values " >> $TMPFILE
for (( i=1; i<=$COUNT; i++ ))
do
          echo -n "$i"
          if [ $i != $COUNT ]
          then
                    echo ","
          fi
done >> $TMPFILE

echo ";" >> $TMPFILE
echo >> $TMPFILE

SEED=$RANDOM

echo "create table rnd$STREAM as select random(DEC(12),$SEED,0,1,$[$SF*1500000]) rnd, i from indices$STREAM;" >> $TMPFILE
echo "COMMIT;" >> $TMPFILE
echo >> $TMPFILE

echo "update rnd$STREAM set rnd=trunc(rnd/8)*32+mod(rnd,8);" >> $TMPFILE
echo "COMMIT;" >> $TMPFILE
echo >> $TMPFILE

echo "create table rndwithln$STREAM as select rnd, i, max(l_linenumber) ln from lineitem, rnd$STREAM where l_orderkey=rnd
group by i,rnd order by i;" >> $TMPFILE
echo "COMMIT;" >> $TMPFILE
echo >> $TMPFILE

SEED=$RANDOM
echo "update rndwithln$STREAM set ln=random( dec(10), $SEED, 0, 1, ln );" >> $TMPFILE
echo "COMMIT;" >> $TMPFILE
echo >> $TMPFILE

SEED=$RANDOM
echo  "create table values$STREAM as select rnd O_KEY, ln L_KEY, random( dec(10), $SEED, 0, 1, 100 ) DELTA, i from
rndwithln$STREAM;" >> $TMPFILE
echo "COMMIT;" >> $TMPFILE

echo "drop table indices$STREAM;" >> $TMPFILE
echo "drop table rnd$STREAM;" >> $TMPFILE
echo "drop table rndwithln$STREAM;" >> $TMPFILE
echo "COMMIT;" >> $TMPFILE

echo >> $TMPFILE
echo "select O_KEY,L_KEY,DELTA from values$STREAM order by i;" >> $TMPFILE
echo >> $TMPFILE
echo "drop table values$STREAM;" >> $TMPFILE
echo >> $TMPFILE
echo "COMMIT;" >> $TMPFILE
echo  >> $TMPFILE
echo "exit;" >> $TMPFILE
echo >> $TMPFILE

# execute SQLFILE with exaplus. Filter output so only the values for O_KEY, L_KEY and DELTA remain
$EXAPLUS -h $HOST -p $ROOTPORT -u tpcuser -P tpcuser \
          -f $TMPFILE | grep "  " | grep -v "O_KEY" \
          > $OUTFILE



# ------------------------------------------------------------------------------------------
# store_values_for_orderkeys.sh
# ------------------------------------------------------------------------------------------
#!/bin/bash
# param <roothost> <orderkeysfile> <outfile> <logdir>
# precondition:
#                   - EXASolution must be running and accepting connections on <roothost>:$ROOTPORT
#                   - the path to <outfile> should exist

ROOTHOST="$1"
ORDERKEYS="$2"
OUTFILE="$3"
LOGDIR="$4"

ROOTPORT=26822

ACIDDIR="$HOME/TPCH/ACID"
TMPDIR=$LOGDIR/tmp
```

```
mkdir -p $TMPDIR

echo "OPEN SCHEMA tpc;" > $TMPDIR/tmp_queryfile.sql
echo >> $TMPDIR/tmp_queryfile.sql
echo "SELECT * FROM history ORDER BY h_date_t;" >> $TMPDIR/tmp_queryfile.sql
# echo $TMPDIR/tmp_queryfile.sql
echo "SELECT * FROM lineitem WHERE l_orderkey in (" >> $TMPDIR/tmp_queryfile.sql
# insert the orderkeys as a comma-separated list into the queryfile
cat  $ORDERKEYS  |  sed  "s/[0-9]  /&A/g"  |  sed  "s/A.*$/,/g"  |  tr  -d  '\n'  |  sed  "s/,$//g"  |  sed  "s/,/,\n/g"  >>
$TMPDIR/tmp_queryfile.sql
echo ")" >> $TMPDIR/tmp_queryfile.sql
echo "ORDER BY l_orderkey, l_linenumber;" >> $TMPDIR/tmp_queryfile.sql
echo >> $TMPDIR/tmp_queryfile.sql
echo "SELECT * FROM orders WHERE o_orderkey in (" >> $TMPDIR/tmp_queryfile.sql
# insert the orderkeys as a comma-separated list into the queryfile
cat  $ORDERKEYS  |  sed  "s/[0-9]  /&A/g"  |  sed  "s/A.*$/,/g"  |  tr  -d  '\n'  |  sed  "s/,$//g"  |  sed  "s/,/,\n/g"  >>
$TMPDIR/tmp_queryfile.sql
#cat $ORDERKEYS >> $TMPDIR/tmp_queryfile.sql
echo ")" >> $TMPDIR/tmp_queryfile.sql
echo "ORDER BY o_orderkey;" >> $TMPDIR/tmp_queryfile.sql
echo  >> $TMPDIR/tmp_queryfile.sql
echo "EXIT;" >> $TMPDIR/tmp_queryfile.sql

cat $TMPDIR/tmp_queryfile.sql | $EXAPLUS -h $ROOTHOST -p $ROOTPORT -u sys -P exasol > $OUTFILE
# rm -f $TMPDIR/tmp_queryfile.sql



# ------------------------------------------------------------------------------------------
# acidquery.sh
# ------------------------------------------------------------------------------------------
#!/bin/bash
# params <HOST> <LOGDIR> <OKEY>

HOST=$1
LOGDIR=$2
OKEY=$3

ROOTPORT=26822

TMPFILE=$LOGDIR/query_tmp.sql

rm -f $TMPFILE

echo "set autocommit off;" > $TMPFILE
echo "open schema tpc;" >> $TMPFILE
echo "select sum(trunc(trunc(l_extendedprice*(1-l_discount),2)*(1+l_tax),2))" >> $TMPFILE
echo "from lineitem" >> $TMPFILE
echo "where l_orderkey=$OKEY;" >> $TMPFILE

$EXAPLUS -h $HOST -p $ROOTPORT -u tpcuser -P tpcuser -f $TMPFILE | grep "\(SUM\|--\|  \)"

#rm -r $TMPFILE



# ------------------------------------------------------------------------------------------
# transaction/querystream.h
# ------------------------------------------------------------------------------------------
#ifndef QUERYSTREAM_H
#define QUERYSTREAM_H

#include <string>
#include <iostream>

#include "exaCInterface.h"

/** \brief The maximum length of a column name. */
static const size_t maxColumnNameSize = 40;

/** \brief A structure to describe connection information to a database.
  * Used in QueryStream::connectToServer().
  * \sa QueryStream::connectToServer() */
struct Configuration
{
    /** \brief The hostname of the server. */
    char *host;
    /** \brief The port of the server to connect to. */
    char *port;
    /** \brief The user name used for authentication to the database. */
    char *user;
    /** \brief The password of the user. */
    char *password;
};

/** \brief This exception is thrown when a transaction was automatically rolled back.*/
class rollbackException
{
};

/** \brief A structure used to fetch data from a database.
  * Each database column is represented by one FetchStruct.
  * Only one of the arrays of ODBC-C-datatypes is initialized and filled per column.
  * That one is identified by the c_datatype. */
struct FetchStruct
{
    /** \brief Array for columns of SQLCHAR* type
     *(one large block, easy access by sqlCharArrayAccess). */
    SQLCHAR         *sqlCharArray;
```

```
    /** \brief To quickly access the i-th string in the sqlCharArray array. */
          SQLCHAR*            *sqlCharArrayAccess;
    /** \brief Array for columns of SQLSMALLINT type. */
    SQLSMALLINT     *sqlSmallInt;
    /** \brief Array for columns of SQLUSMALLINT type. */
    SQLUSMALLINT    *sqlUSmallInt;
    /** \brief Array for columns of SQLINTEGER type. */
    SQLINTEGER      *sqlInteger;
    /** \brief Array for columns of SQLUINTEGER type. */
    SQLUINTEGER     *sqlUInteger;
    /** \brief Array for columns of SQLBIGINT type. */
    SQLBIGINT       *sqlBigInt;
    /** \brief Array for columns of SQLUBIGINT type. */
    SQLUBIGINT      *sqlUBigInt;
    /** \brief Array for columns of SQLREAL type. */
    SQLREAL         *sqlReal;
    /** \brief Array for columns of SQLDOUBLE type. */
    SQLDOUBLE       *sqlDouble;
    /** \brief Array for columns of SQLCHAR type. */
    SQLCHAR         *sqlChar;
    /** \brief Array for columns of SQLSCHAR type. */
    SQLSCHAR        *sqlSChar;
    /** \brief Array for columns of SQL_DATE_STRUCT type. */
    SQL_DATE_STRUCT *sqlDate;
    /** \brief Array for columns of SQL_NUMERIC_STRUCT type. */
    SQL_NUMERIC_STRUCT *sqlNumeric;

    /** \brief The length of a returned data and an indicator whether it is NULL. */
    SQLLEN *resultLen;

    /** \brief The name of the column. */
    SQLCHAR columnName[maxColumnNameSize];
    /** \brief The length of the column name. */
    SQLSMALLINT nameLength;
    /** \brief The SQL datatype ID of the column. */
    SQLSMALLINT datatype;
    /** \brief The ODBC C datatype ID of the column.
      * Might be set by QueryStream::find_ODBC_C_Type( SQLSMALLINT sqlType ) */
    SQLSMALLINT c_datatype;
    /** \brief The size of the column. */
    SQLUINTEGER columnSize;
    /** \brief The number of decimal digits of the column. */
    SQLSMALLINT decimalDigits;
    /** \brief Whether the column allows NULL values. */
    SQLSMALLINT nullable;
};

/** \brief The class represents a query stream to a server.
  * This query stream is slightly different from that one used by the Performance test
  * since it is for implementing the ACID-Transaction for which queries are only
  * executed to get (or change) single lines whose values must be accessible since
  * they will be changed.
  *
  * The execute(...) function was slightly changed to fetch at most two rows and not output or delete
  * the result set. Furthermore a new function freeFetchArray() was introduced to allow the user of
  * this class to free the fetchMe array. Last but not least the fetchMe array of fetched rows is
  * publicly accessable by several getDatatype(colum) functions to allow reading of returned data.
  *
  * In addition to that the mapping of C-Datatypes to SQL-Types might have changed slightly to ensure
  * mapping of some columns to numerical C-Types. This affects find_ODBC_C_Type(...). */
class QueryStream
{
public:
    /** \brief Constructor.
      * \param streamId The identifier used to identify this query stream.
      * \param logfile The stream to be used for logging. */
    QueryStream( std::string streamId, std::ostream* logfile = &std::cout );
    /** \brief Tries to establish a connection to a server.
      * \param configuration The connection and login information for the server.
      * \return true if the connection could be established, false otherwise. */
    bool connectToServer( Configuration configuration );
    /** \brief Executes a given statement or query.
      * \param statement The text of the statement or query to execute.
      * \param fetchSize The number of rows to fetch with a single fetch command
      * (bigger fetch size might reduce overhead, but increases memory usage).
      * \return The number of rows in resultset (-2 (error), -1 (no resultset), 0,1 or 2 (means more than one)).
      */
    int execute( std::string statement );
    /** \brief Closes the connection to the server.
      * \return true if the connection was closed successfully, false if errors occured. */
    bool disconnectFromServer();
          /** \brief frees the memory allocated to store the resultset.
            * Call this exactly if execute did not return -1. */
          void freeFetchArray();

          int getInt( size_t columnNumber );

          double getDouble( size_t columnNumber );
protected:
    /** \brief The maximum size of a string. */
    static const size_t maxStringSize  = 200;
    /** \brief The number of rows to fetch in a single fetch command. */
    SQLINTEGER fetchBlockSize;
          /** \brief The number to use for fetchBlockSize if it is not
        * given as argument to execute(). */
          static const size_t standardFetchBlockSize = 100;

    /** \brief Binds a column to an entry in the fetchMe array.
      * This must be done before fetching a resultset.
      * This function arranges memory allocation and binding function.
      * \param colNo The number of the resultset column to bind to.
```

```
      * \param localColNo The index in the fetchMe array the resultset data shall be stored.
      * \param datatype The SQL datatype of the column to bind to. */
    void bindColumn( int colNo, int localColNo, SQLSMALLINT datatype );

    /** \brief Allocates memory for the buffers needed in binding resultset columns.
      * The array corresponding to the given datatype of the given index
      * of the fetchMe array is initialized to the size of fetchBlockSize.
      * If the datatype is a string datatype, the string buffer will have the
      * size maxStringSize.
      * \param localColNo The index in the fetchMe array the data shall be allocated for.
      * \param datatype The ODBC-C-datatype memory shall be allocated for. */
    void allocateMemoryForBind( int localColNo, SQLSMALLINT datatype );

    /** \brief Maps an SQL-datatype-ID to a ODBC-C-datatype-ID to which it can be converted.
      * This is needed for binding columns and fetching results from a remote database.
      * \param sqlType The SQL-datatype-ID whose corresonding ODBC-C-datatype-ID is wanted.
      * \return The ODBC-C-datatype-ID for the given SQL-datatype-ID */
    SQLSMALLINT find_ODBC_C_Type( SQLSMALLINT sqlType );

    /** \brief Format resultset header for output. */
    void outputHeader();

    /** \brief Formats fetched rows for output.
      * The data in fetchMe is printed for each column and all fetched rows. */
    void outputFetchedRows();

    /** \brief The stream identifier. */
    std::string myID;

    /** \brief The environment handle used. */
    SQLHENV  henv;
    /** \brief The connection handle used. */
    SQLHDBC  hdbc;
    /** \brief The only statement handle used. */
    SQLHSTMT hstmt;

    /** \brief The structure used to fetch result sets of queries. */
    FetchStruct *fetchMe;

    /** \brief The number of rows in a result set. */
    SQLSMALLINT numResultCols;

    /** \brief The number of rows fetched by last fetch command. */
    SQLUINTEGER rowsFetched;

    /** \brief The stream used for logging results.*/
    std::ostream* logfile;

private:
         /** \brief Parses the return code of an statement and gives additional information on error. */
         SQLRETURN execCritical(SQLRETURN retcode, SQLSMALLINT handle_type = 0, SQLHANDLE handle = 0);
};

#endif // QUERYSTREAM_H



# ------------------------------------------------------------------------------------
# transaction/querystream.cpp
# ------------------------------------------------------------------------------------
#include "querystream.h"
#include <iostream>

//#define DEBUG
#ifdef DEBUG
#define DEBUGOUT( X ) cout << "DEBUG:" << X
#else
#define DEBUGOUT( X )
#endif

#define LOG *logfile

using std::cout;
using std::endl;
using std::cerr;

#include <string>
using std::string;

SQLRETURN QueryStream::execCritical(SQLRETURN retcode, SQLSMALLINT handle_type, SQLHANDLE handle)
{
    switch(retcode)
    {
    case SQL_INVALID_HANDLE:
                cout << "invalid handle" << endl;
        throw "ERROR use of invalid handle!\n";
        break;
    case SQL_ERROR:
         {
                SQLCHAR err[1000];
                SQLCHAR state[6];
                SQLINTEGER native;
                SQLSMALLINT len;
                SQLSMALLINT i=1;
       while ( handle !=0 && EXAGetDiagRec(handle_type, handle, i, state, &native,err,1000,&len) == SQL_SUCCESS )
                {
                        cout << state << ":" << err << endl;
                        i++;
                        if ( string( (const char*) err).find( "GlobalTransactionRollback") )
                        {
                                throw rollbackException();
```

```
                        }
                }
                cout << "SQL error" << endl;
        throw "An error occured while executing command!\n";
        break;
          }
          case SQL_SUCCESS_WITH_INFO:
                cout << "Problem: Has Info, which should not be" << endl;
                throw "Problem: Has Info, which should not be!\n";
          break;
          case SQL_STILL_EXECUTING:
                cout << "Still executing. This should not be should only";
                cout << "return on successful execution!" << endl;
                throw "Unexpectedly still executing!\n";
                break;
          case SQL_NO_DATA:
          case SQL_SUCCESS:
        break;
          default:
                // should not reach here...
                cout << "unexpected return code: " << retcode << endl;
                throw "unexpected return code!\n";
                break;
    }
    return retcode;
}

QueryStream::QueryStream( std::string streamId, std::ostream* logfile_ )
{
    myID  = streamId;
    henv  = NULL;
    hdbc  = NULL;
    hstmt = NULL;
    logfile = logfile_;
}

bool QueryStream::connectToServer( Configuration cfg )
{
    char localeAttr[]="deu";
    try
    {
        // Allocating environment handle
        execCritical(EXAAllocHandle(SQL_HANDLE_ENV, NULL, &henv));
        // Setting up environment to german localization
        execCritical(EXASetEnvAttr(henv, EXA_ENV_ATTR_SET_LOCALE_ALL, localeAttr, SQL_NTS));
        // Allocating handle for connection
        execCritical(EXAAllocHandle(SQL_HANDLE_DBC, henv, &hdbc));
        // Turning autocommit off... is this necessary?
        execCritical(EXASetConnectAttr(hdbc, SQL_ATTR_AUTOCOMMIT, (void*)SQL_AUTOCOMMIT_OFF, SQL_NTS) );
        // Setting up clientname
        execCritical(EXASetConnectAttr(hdbc, EXA_CLIENT_NAME, (void*)myID.c_str(), SQL_NTS));
        // Connect to server
        execCritical(EXAServerConnect(hdbc, cfg.host, SQL_NTS, cfg.port, SQL_NTS, cfg.user, SQL_NTS, cfg.password, SQL_NTS));
    }
    catch(...)
    {
        // might be nice to clean up handles
        return false;
    }
    return true;
}

int QueryStream::execute( std::string statement )
{
    // the number of rows in the resultSet
    int resultSetSize = -1;

    try
    {
        // put the statement in the logfile
        LOG << statement << endl;
        // Allocating statement handle
        execCritical(EXAAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt), SQL_HANDLE_DBC, hdbc);
        SQLCHAR* statementtext = (SQLCHAR*) statement.c_str();
        // execute the statement / query
        execCritical(EXAExecDirect(hstmt, statementtext, statement.size() ), SQL_HANDLE_STMT, hstmt );
        // get the affected rows (only insert, update, delete)
        SQLINTEGER affectedrows;
        execCritical(EXARowCount(hstmt,&affectedrows), SQL_HANDLE_STMT, hstmt);
        // get the number of columns in the resultset (if any)
        execCritical(EXANumResultCols(hstmt,&numResultCols), SQL_HANDLE_STMT, hstmt);
        // no result columns means no query
        if ( numResultCols <= 0 )
            LOG << "affected rows: " << affectedrows << "\n\n";

        // if a resultset was constructed, retrieve and print it
        if ( numResultCols > 0 )
        // a resultset was constructed
        {
                            resultSetSize = 0;
            LOG << '\n';
            // setting number of rows to fetch with each fetch
            SQLUINTEGER fetchsize = 2; // This is to store the result set and be able to ensure only one line was received
                            fetchBlockSize = 2;
            execCritical(EXASetStmtAttr(hstmt, SQL_ATTR_ROW_ARRAY_SIZE,
                                        (SQLPOINTER) fetchsize , 0 ), SQL_HANDLE_STMT, hstmt);
            //cout << "Fetch block size (SQL_ATTR_ROW_ARRAY_SIZE) set to " << fetchsize << endl;
            // setting position to store the fetch size for a query
            execCritical(EXASetStmtAttr(hstmt, SQL_ATTR_ROWS_FETCHED_PTR,
                                        (SQLPOINTER) &rowsFetched, 0), SQL_HANDLE_STMT, hstmt);
            // allocate the fetch struct to be large enough for the columns
```

```
            fetchMe = new FetchStruct[numResultCols];
            // set all pointers to 0 to make it easy to free the correct memory later
            for ( int i=0; i<numResultCols; i++ )
            {
                fetchMe[i].sqlCharArray = 0;
                                        fetchMe[i].sqlCharArrayAccess = 0;
                fetchMe[i].sqlSmallInt  = 0;
                fetchMe[i].sqlUSmallInt = 0;
                fetchMe[i].sqlInteger   = 0;
                fetchMe[i].sqlUInteger  = 0;
                fetchMe[i].sqlBigInt    = 0;
                fetchMe[i].sqlUBigInt   = 0;
                fetchMe[i].sqlReal      = 0;
                fetchMe[i].sqlDouble    = 0;
                fetchMe[i].sqlChar      = 0;
                fetchMe[i].sqlSChar     = 0;
                fetchMe[i].sqlDate      = 0;
                fetchMe[i].sqlNumeric   = 0;
                fetchMe[i].resultLen    = 0;
            }
            execCritical(EXASetStmtAttr(hstmt, SQL_ATTR_ROWS_FETCHED_PTR,
                                    (SQLPOINTER) &rowsFetched, 0), SQL_HANDLE_STMT, hstmt);
            // determine the datatype of the column and additional information.
            for ( SQLSMALLINT colNr=0; colNr<numResultCols; colNr++ )
            {
                // Get column description for the column with number colNr
                EXADescribeCol( hstmt, colNr+1, fetchMe[colNr].columnName,
                                maxColumnNameSize, &(fetchMe[colNr].nameLength),
                                &(fetchMe[colNr].datatype), &(fetchMe[colNr].columnSize),
                                &(fetchMe[colNr].decimalDigits), &(fetchMe[colNr].nullable) );
                fetchMe[colNr].c_datatype = find_ODBC_C_Type( fetchMe[colNr].datatype );

                // bind the column to an entry in the fetchMe array
                bindColumn( colNr+1, colNr, fetchMe[colNr].c_datatype );
            }
            // output the resultset header
            outputHeader();
            // fetch the columns
            do
            {
                // fetch columns
                execCritical(EXAFetch( hstmt ), SQL_HANDLE_STMT, hstmt);
                // output columns
                outputFetchedRows();
                // correct the result set size
                resultSetSize += rowsFetched;
            } while ( rowsFetched == fetchBlockSize );
            // add the total resultset size to the logfile
            LOG << "\nresultset consists of " << resultSetSize << " rows in " << numResultCols << " columns\n\n";
        }
    }
          catch(rollbackException)
          {
        // free the handle for the statement again
        cout << "\nROLLBACK EXECUTING STATEMENT:\n" << statement << '\n' << endl;
        execCritical(EXAFreeHandle( SQL_HANDLE_STMT, hstmt ), SQL_HANDLE_STMT, hstmt);
                    return -3;
          }
          catch( char * c )
          {
        LOG << "ERROR EXECUTING STATEMENT\n\n";
        cout << "\nERROR EXECUTING STATEMENT:\n" << statement << '\n' << endl;
                    cout << c << endl;
        execCritical(EXAFreeHandle( SQL_HANDLE_STMT, hstmt ), SQL_HANDLE_STMT, hstmt);
                    return -2;
          }
    catch(...)
    {
        LOG << "ERROR EXECUTING STATEMENT\n\n";
        cout << "\nERROR EXECUTING STATEMENT:\n" << statement << '\n' << endl;
        execCritical(EXAFreeHandle( SQL_HANDLE_STMT, hstmt ), SQL_HANDLE_STMT, hstmt);
                    return -2;
    }
    execCritical(EXAFreeHandle( SQL_HANDLE_STMT, hstmt ), SQL_HANDLE_STMT, hstmt);
          return resultSetSize;
}

void QueryStream::freeFetchArray()
{
          for ( int i=0; i<numResultCols; i++ )
    {
        if ( fetchMe[i].sqlCharArray != 0 )
        {
            delete[] fetchMe[i].sqlCharArray;
            delete[] fetchMe[i].sqlCharArrayAccess;
        }
        if ( fetchMe[i].sqlSmallInt  != 0 ) delete[] fetchMe[i].sqlSmallInt;
        if ( fetchMe[i].sqlUSmallInt != 0 ) delete[] fetchMe[i].sqlUSmallInt;
        if ( fetchMe[i].sqlInteger   != 0 ) delete[] fetchMe[i].sqlInteger;
        if ( fetchMe[i].sqlUInteger  != 0 ) delete[] fetchMe[i].sqlUInteger;
        if ( fetchMe[i].sqlBigInt    != 0 ) delete[] fetchMe[i].sqlBigInt;
        if ( fetchMe[i].sqlUBigInt   != 0 ) delete[] fetchMe[i].sqlUBigInt;
        if ( fetchMe[i].sqlReal      != 0 ) delete[] fetchMe[i].sqlReal;
        if ( fetchMe[i].sqlDouble    != 0 ) delete[] fetchMe[i].sqlDouble;
        if ( fetchMe[i].sqlChar      != 0 ) delete[] fetchMe[i].sqlChar;
        if ( fetchMe[i].sqlSChar     != 0 ) delete[] fetchMe[i].sqlSChar;
        if ( fetchMe[i].sqlDate      != 0 ) delete[] fetchMe[i].sqlDate;
        if ( fetchMe[i].sqlNumeric   != 0 ) delete[] fetchMe[i].sqlNumeric;
        if ( fetchMe[i].resultLen    != 0 ) delete[] fetchMe[i].resultLen;
    }
    delete[] fetchMe;
```

```
}


int QueryStream::getInt( size_t columnNumber )
{
          size_t i = columnNumber;
          if ( i >= numResultCols )
          {
                    cout << "column out of range" << endl;
                    throw "column out of range";
          }

          if ( fetchMe[i].sqlCharArray != 0 ) return atoi( (char*) fetchMe[i].sqlCharArrayAccess[0] );

          if ( fetchMe[i].sqlSmallInt  != 0 ) return fetchMe[i].sqlSmallInt[0];
    if ( fetchMe[i].sqlUSmallInt != 0 ) return fetchMe[i].sqlUSmallInt[0];
    if ( fetchMe[i].sqlInteger   != 0 ) return fetchMe[i].sqlInteger[0];
    if ( fetchMe[i].sqlUInteger  != 0 ) return fetchMe[i].sqlUInteger[0];
    if ( fetchMe[i].sqlBigInt    != 0 ) return fetchMe[i].sqlBigInt[0];
    if ( fetchMe[i].sqlUBigInt   != 0 ) return fetchMe[i].sqlUBigInt[0];
    if ( fetchMe[i].sqlReal      != 0 ) return (int) fetchMe[i].sqlReal[0];
    if ( fetchMe[i].sqlDouble    != 0 ) return (int) fetchMe[i].sqlDouble[0];
    if ( fetchMe[i].sqlChar      != 0 ) return fetchMe[i].sqlChar[0];
    if ( fetchMe[i].sqlSChar     != 0 ) return fetchMe[i].sqlSChar[0];

          cout << "cannot cast datatype to Int" << endl;
          throw "cannot cast datatype to Int";
}

double QueryStream::getDouble( size_t columnNumber )
{
          size_t i = columnNumber;
          if ( i >= numResultCols )
          {
                    cout << "column out of range" << endl;
                    throw "column out of range";
          }

          if ( fetchMe[i].sqlCharArray != 0 ) return atof( (char*) fetchMe[i].sqlCharArrayAccess[0] );

          if ( fetchMe[i].sqlSmallInt  != 0 ) return fetchMe[i].sqlSmallInt[0];
    if ( fetchMe[i].sqlUSmallInt != 0 ) return fetchMe[i].sqlUSmallInt[0];
    if ( fetchMe[i].sqlInteger   != 0 ) return fetchMe[i].sqlInteger[0];
    if ( fetchMe[i].sqlUInteger  != 0 ) return fetchMe[i].sqlUInteger[0];
    if ( fetchMe[i].sqlBigInt    != 0 ) return fetchMe[i].sqlBigInt[0];
    if ( fetchMe[i].sqlUBigInt   != 0 ) return fetchMe[i].sqlUBigInt[0];
    if ( fetchMe[i].sqlReal      != 0 ) return fetchMe[i].sqlReal[0];
    if ( fetchMe[i].sqlDouble    != 0 ) return fetchMe[i].sqlDouble[0];
    if ( fetchMe[i].sqlChar      != 0 ) return fetchMe[i].sqlChar[0];
    if ( fetchMe[i].sqlSChar     != 0 ) return fetchMe[i].sqlSChar[0];

          cout << "cannot cast datatype to Int" << endl;
          throw "cannot cast datatype to Int";

}

bool QueryStream::disconnectFromServer()
{
    try
    {
        // disconnecting from server
        execCritical(EXADisconnect(hdbc));
        // freeing connection handle
        execCritical(EXAFreeHandle(SQL_HANDLE_DBC, hdbc));
        // freeing environment handle
        execCritical(EXAFreeHandle(SQL_HANDLE_ENV, henv));
    }
    catch(...)
    {
        cout << "Error while disconnecting from server" << endl;
        return false;
    }
    return true;
}

void QueryStream::outputHeader()
{
    int headerLength = 1;
    LOG << "|";
    for ( size_t col=0; col<numResultCols; col++ )
    {
        LOG << fetchMe[col].columnName << "|";
        headerLength += fetchMe[col].nameLength + 1;
    }
    LOG << '\n';
    for ( int i=0; i<headerLength; i++ )
        LOG << '=';
    LOG << '\n';
          (LOG).flush();
}

void QueryStream::outputFetchedRows()
{
    for ( size_t rows=0; rows<rowsFetched; rows++ )
    {
        LOG << "|";
        for ( size_t cols=0; cols<numResultCols; cols++ )
        {
            if ( fetchMe[cols].resultLen[rows] == SQL_NULL_DATA )
            {
                LOG << "|";
```

```
    continue;
}
switch( fetchMe[cols].c_datatype )
{
    // string data
    case SQL_C_CHAR:
    case SQL_C_BINARY:
    //case SQL_C_XML:
    //case SQL_C_VARBOOKMARK:
        LOG << fetchMe[cols].sqlCharArrayAccess[rows] << "|";
    break;

    // small int
    case SQL_C_SSHORT:
        LOG << fetchMe[cols].sqlSmallInt[rows] << "|";
    break;

    // unsigned small int
    case SQL_C_USHORT:
        LOG << fetchMe[cols].sqlUSmallInt[rows] << "|";
    break;

    // long int
    case SQL_C_SLONG:
        LOG << fetchMe[cols].sqlInteger[rows] << "|";
    break;

    // unsigned long int
    case SQL_C_ULONG:
        LOG << fetchMe[cols].sqlUInteger[rows] << "|";
    break;

    // float
    case SQL_C_FLOAT:
        LOG << fetchMe[cols].sqlReal[rows] << "|";
    break;

    // double
    case SQL_C_DOUBLE:
        LOG << fetchMe[cols].sqlDouble[rows] << "|";
    break;

    // unsigned char
    case SQL_C_BIT:
    case SQL_C_UTINYINT:
        LOG << fetchMe[cols].sqlChar[rows] << "|";
    break;

    // signed char
    case SQL_C_STINYINT:
        LOG << fetchMe[cols].sqlSChar[rows] << "|";
    break;

    // big int
    case SQL_C_SBIGINT:
        LOG << fetchMe[cols].sqlBigInt[rows] << "|";
    break;

    // unsigned big int
    case SQL_C_UBIGINT:
        LOG << fetchMe[cols].sqlUBigInt[rows] << "|";
    break;

    // date
    case SQL_C_TYPE_DATE:
    {
        // format date to YYYY-MM-DD for output
        if ( fetchMe[cols].sqlDate[rows].year < 1000 )
            LOG << "0";
        if ( fetchMe[cols].sqlDate[rows].year < 100 )
            LOG << "0";
        if ( fetchMe[cols].sqlDate[rows].year < 10 )
            LOG << "0";
        LOG << fetchMe[cols].sqlDate[rows].year << "-";

        if ( fetchMe[cols].sqlDate[rows].month < 10 )
            LOG << "0";
        LOG << fetchMe[cols].sqlDate[rows].month << "-";

        if ( fetchMe[cols].sqlDate[rows].day < 10 )
            LOG << "0";
        LOG << fetchMe[cols].sqlDate[rows].day << "|";
    }
    break;

    // numeric, i.e. decimal
                          // not used since it is eaier by far to map decimals to doubles
    case SQL_C_NUMERIC:
    {
        // format decimal for output
        int scale = fetchMe[cols].sqlNumeric[rows].scale;
        char text[3*SQL_MAX_NUMERIC_LEN+2+scale];
        int pos_T = 3*SQL_MAX_NUMERIC_LEN+scale;
        int pos_N = SQL_MAX_NUMERIC_LEN-1;
        text[3*SQL_MAX_NUMERIC_LEN+1+scale] = '\0';
        //int add = 0;
        //if ( fetchMe[cols].sqlNumeric[rows].sign == 0 )
        //    add=1;
        while ( fetchMe[cols].sqlNumeric[rows].val[pos_N] == 0 && pos_N > 0)
            pos_N--;
```

```
                    while ( pos_N >= 0 )
                    {
                        int current = fetchMe[cols].sqlNumeric[rows].val[pos_N];
                        int newVal = 0;
                        int currentPos = pos_N;
                        while ( currentPos >= 0 )
                        {
                            //if ( currentPos == 0 )
                            //{
                            //    current += add;
                            //    add = 0;
                            //}
                            if ( current < 10 )
                            {

                                fetchMe[cols].sqlNumeric[rows].val[currentPos] = newVal;
                                currentPos--;

                                if ( currentPos >= 0 )
                                {
                                    current = current * 256 + fetchMe[cols].sqlNumeric[rows].val[currentPos];
                                    newVal = 0;
                                }
                            }
                            else
                            {
                                newVal = current / 10;
                                current = current % 10;
                            }
                        }
                        if ( fetchMe[cols].sqlNumeric[rows].val[pos_N] == 0 )
                        {
                            pos_N--;
                        }
                        text[pos_T--] = '0' + current;
                        if ( scale == 1 )
                        {
                            text[pos_T--] = '.';
                        }
                        scale--;
                    }
                    if ( scale==0)
                        text[pos_T--] = '0';
                    while ( scale > 0 )
                    {
                        text[pos_T--] = '0';
                        if ( scale == 1 )
                        {
                            text[pos_T--] = '.';
                            text[pos_T--] = '0';
                        }
                        scale--;
                    }

                    if ( fetchMe[cols].sqlNumeric[rows].sign == 0 )
                        text[pos_T--] = '-';
                    LOG << text+pos_T+1 << "|";
                }
                break;

                // other datatype
                default:
                    cout << "output: datatype " << fetchMe[cols].c_datatype << " not supported yet" << endl;
                    throw fetchMe[cols].c_datatype;
                break;
            }
        }
        LOG << '\n';
                (LOG).flush();
    }
}

void QueryStream::bindColumn( int colNo, int localColNo, SQLSMALLINT c_datatype )
{
    // allocate memory for the column to bind
    allocateMemoryForBind( localColNo, c_datatype );
    // bind the column
    switch ( c_datatype )
    {
        // string data
        case SQL_C_CHAR:
        case SQL_C_BINARY:
        //case SQL_C_XML:
        //case SQL_C_VARBOOKMARK:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlCharArray,
                    maxStringSize*sizeof(SQLCHAR), fetchMe[localColNo].resultLen));
        break;

        // small int
        case SQL_C_SSHORT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlSmallInt,
                    sizeof(SQLSMALLINT), fetchMe[localColNo].resultLen));
        break;

        // unsigned small int
        case SQL_C_USHORT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlUSmallInt,
                    sizeof(SQLUSMALLINT), fetchMe[localColNo].resultLen));
```

```
        break;

        // long int
        case SQL_C_SLONG:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlInteger,
                    sizeof(SQLINTEGER), fetchMe[localColNo].resultLen));
        break;

        // unsigned long int
        case SQL_C_ULONG:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlUInteger,
                    sizeof(SQLUINTEGER), fetchMe[localColNo].resultLen));
        break;

        // float
        case SQL_C_FLOAT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlReal,
                    sizeof(SQLREAL), fetchMe[localColNo].resultLen));
        break;

        // double
        case SQL_C_DOUBLE:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlDouble,
                    sizeof(SQLDOUBLE), fetchMe[localColNo].resultLen));
        break;

        // unsigned char
        case SQL_C_BIT:
        case SQL_C_UTINYINT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlChar,
                    sizeof(SQLCHAR), fetchMe[localColNo].resultLen));
        break;

        // signed char
        case SQL_C_STINYINT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlSmallInt,
                    sizeof(SQLSCHAR), fetchMe[localColNo].resultLen));
        break;

        // big int
        case SQL_C_SBIGINT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlBigInt,
                    sizeof(SQLBIGINT), fetchMe[localColNo].resultLen));
        break;

        // unsigned big int
        case SQL_C_UBIGINT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlUBigInt,
                    sizeof(SQLUBIGINT), fetchMe[localColNo].resultLen));
        break;

        // date
        case SQL_C_TYPE_DATE:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlDate,
                    sizeof(SQL_DATE_STRUCT), fetchMe[localColNo].resultLen));
        break;

        // numeric, i.e. decimal
        case SQL_C_NUMERIC:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlNumeric,
                    sizeof(SQL_NUMERIC_STRUCT), fetchMe[localColNo].resultLen));
        break;

        // other datatype
        default:
            throw c_datatype;
        break;
    }
}

void QueryStream::allocateMemoryForBind( int localColNo, SQLSMALLINT datatype )
{
    // allocate memory for resultLen field used to indicate NULL values (among others)
    fetchMe[localColNo].resultLen = new SQLLEN[fetchBlockSize];

    // allocate memory depending on the column datatype
    switch ( datatype )
    {
        // string data
        case SQL_C_CHAR:
        case SQL_C_BINARY:
        //case SQL_C_XML:
        //case SQL_C_VARBOOKMARK:
            fetchMe[localColNo].sqlCharArray = new SQLCHAR[fetchBlockSize*maxStringSize];
                        fetchMe[localColNo].sqlCharArrayAccess = new SQLCHAR*[fetchBlockSize];
            for ( size_t i=0; i<fetchBlockSize; i++ )
            {
                fetchMe[localColNo].sqlCharArrayAccess[i] = fetchMe[localColNo].sqlCharArray+i*maxStringSize;
            }
        break;
```

```
        // small int
        case SQL_C_SSHORT:
            fetchMe[localColNo].sqlSmallInt = new SQLSMALLINT[fetchBlockSize];
        break;

        // unsigned small int
        case SQL_C_USHORT:
            fetchMe[localColNo].sqlUSmallInt = new SQLUSMALLINT[fetchBlockSize];
        break;

        // long int
        case SQL_C_SLONG:
            fetchMe[localColNo].sqlInteger = new SQLINTEGER[fetchBlockSize];
        break;

        // unsigned long int
        case SQL_C_ULONG:
            fetchMe[localColNo].sqlUInteger = new SQLUINTEGER[fetchBlockSize];
        break;

        // float
        case SQL_C_FLOAT:
            fetchMe[localColNo].sqlReal = new SQLREAL[fetchBlockSize];
        break;

        // double
        case SQL_C_DOUBLE:
            fetchMe[localColNo].sqlDouble = new SQLDOUBLE[fetchBlockSize];
        break;

        // unsigned char
        case SQL_C_BIT:
        case SQL_C_UTINYINT:
            fetchMe[localColNo].sqlChar = new SQLCHAR[fetchBlockSize];
        break;

        // signed char
        case SQL_C_STINYINT:
            fetchMe[localColNo].sqlSChar = new SQLSCHAR[fetchBlockSize];
        break;

        // big int
        case SQL_C_SBIGINT:
            fetchMe[localColNo].sqlBigInt = new SQLBIGINT[fetchBlockSize];
        break;

        // unsigned big int
        case SQL_C_UBIGINT:
            fetchMe[localColNo].sqlUBigInt = new SQLUBIGINT[fetchBlockSize];
        break;

        // date
        case SQL_C_TYPE_DATE:
            fetchMe[localColNo].sqlDate = new SQL_DATE_STRUCT[fetchBlockSize];
        break;

        // numeric, i.e. decimal
        case SQL_C_NUMERIC:
            fetchMe[localColNo].sqlNumeric = new SQL_NUMERIC_STRUCT[fetchBlockSize];
        break;

        // other datatype
        default:
            cout << "memory allocation: datatype " << datatype << " not supported yet" << endl;
            throw datatype;
        break;
    }
}

SQLSMALLINT QueryStream::find_ODBC_C_Type( SQLSMALLINT sqlType )
{
    switch( sqlType )
    {
        case SQL_CHAR:
        case SQL_VARCHAR:
        case SQL_LONGVARCHAR:
        case SQL_WCHAR:
        case SQL_WVARCHAR:
        case SQL_WLONGVARCHAR:
            return SQL_C_CHAR;
        break;

        case SQL_DECIMAL:
        case SQL_NUMERIC:
            //return SQL_C_NUMERIC;
            return SQL_C_DOUBLE;
            //return SQL_C_CHAR;
        break;

        case SQL_SMALLINT:
            return SQL_C_SSHORT;
        break;

        case SQL_INTEGER:
            return SQL_C_SLONG;
        break;

        case SQL_REAL:
        case SQL_FLOAT:
        case SQL_DOUBLE:
            return SQL_C_DOUBLE;
```

```
        break;

        case SQL_BIT:
            return SQL_C_BIT;
        break;

        case SQL_TINYINT:
            return SQL_C_STINYINT;
        break;

        case SQL_BIGINT:
            return SQL_C_SBIGINT;
        break;

        case SQL_BINARY:
        case SQL_VARBINARY:
        case SQL_LONGVARBINARY:
            return SQL_C_BINARY;
        break;

        case SQL_TYPE_DATE:
            return SQL_C_TYPE_DATE;
        break;

        default:
            cout << "unsupported datatype, don't know corresponding C type: " << sqlType << endl;
            throw sqlType;
        break;
    }
}
```

```
# ------------------------------------------------------------------------------------
# create_history_table.sql
# ------------------------------------------------------------------------------------
OPEN SCHEMA tpc;

-- KEYs     DEC(13)
-- INTEGERs DEC(10)
-- DECIMALs DEC(12,2)

CREATE OR REPLACE TABLE HISTORY( H_P_KEY DEC(13), H_S_KEY DEC(13), H_O_KEY DEC(13), H_L_KEY DEC(10), H_DELTA DEC(10),
H_DATE_T TIMESTAMP );

COMMIT;
exit;
```

```
# ------------------------------------------------------------------------------------
# create_history_table2.sql
# ------------------------------------------------------------------------------------
OPEN SCHEMA tpc;

-- KEYs     DEC(13)
-- INTEGERs DEC(10)
-- DECIMALs DEC(12,2)

CREATE OR REPLACE TABLE HISTORY2( H_P_KEY DEC(13), H_S_KEY DEC(13), H_O_KEY DEC(13), H_L_KEY DEC(10), H_DELTA DEC(10),
H_DATE_T TIMESTAMP );

COMMIT;
exit;
```

```
# ------------------------------------------------------------------------------------
# clear_history.sql
# ------------------------------------------------------------------------------------
set autocommit off;
open schema tpc;
delete * from history;
commit;
exit;
```

```
# ------------------------------------------------------------------------------------
# clear_history2.sql
# ------------------------------------------------------------------------------------
set autocommit off;
open schema tpc;
delete * from history2;
commit;
exit;
```

```
# ------------------------------------------------------------------------------------
# store_history.sql
# ------------------------------------------------------------------------------------
set autocommit off;
open schema tpc;
select * from history;
rollback;
exit;
```

```
# ------------------------------------------------------------------------------
# store_history2.sql
# ------------------------------------------------------------------------------
set autocommit off;
open schema tpc;
select * from history2;
rollback;
exit;
```

Appendix D: Query Text and Query Output

```
******************
* TPC-H Query 1 0 *
******************
-- Minor modification - date arithmetic for days ( 2.2.3.3 c) )
select
          l_returnflag,
          l_linestatus,
          sum(l_quantity) as sum_qty,
          sum(l_extendedprice) as sum_base_price,
          sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
          sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
          avg(l_quantity) as avg_qty,
          avg(l_extendedprice) as avg_price,
          avg(l_discount) as avg_disc,
          count(*) as count_order
from
          lineitem
where
          l_shipdate <= date '1998-12-01' - 90
group by
          l_returnflag,
          l_linestatus
order by
          l_returnflag,
          l_linestatus;
```

```
|L_RETURNFLAG|L_LINESTATUS|SUM_QTY|SUM_BASE_PRICE|SUM_DISC_PRICE|SUM_CHARGE|AVG_QTY|AVG_PRICE|AVG_DISC|COUNT_ORDER|
================================================================================================================
|A|F|37734107.00|56586554400.73|53758257134.87|55909065222.83|25.52|38273.13|0.05|1478493.00|
|N|F|991417.00|1487504710.38|1413082168.05|1469649223.19|25.52|38284.47|0.05|38854.00|
|N|O|74476040.00|111701729697.74|106118230307.61|110367043872.50|25.50|38249.12|0.05|2920374.00|
|R|F|37719753.00|56568041380.90|53741292684.60|55889619119.83|25.51|38250.85|0.05|1478870.00|

resultset consists of 4 rows in 10 columns
```

```
******************
* TPC-H Query 2 0 *
******************
-- Minor modification - result set limit ( 2.1.2.9.3 )
select
          s_acctbal,
          s_name,
          n_name,
          p_partkey,
          p_mfgr,
          s_address,
          s_phone,
          s_comment
from
          part,
          supplier,
          partsupp,
          nation,
          region
where
          p_partkey = ps_partkey
          and s_suppkey = ps_suppkey
          and p_size = 15
          and p_type like '%BRASS'
          and s_nationkey = n_nationkey
          and n_regionkey = r_regionkey
          and r_name = 'EUROPE'
          and ps_supplycost = (
                    select
                              min(ps_supplycost)
                    from
                              partsupp,
                              supplier,
                              nation,
                              region
                    where
                              p_partkey = ps_partkey
                              and s_suppkey = ps_suppkey
                              and s_nationkey = n_nationkey
                              and n_regionkey = r_regionkey
                              and r_name = 'EUROPE'
          )
order by
          s_acctbal desc,
          n_name,
          s_name,
          p_partkey
LIMIT 100
;
```

```
|S_ACCTBAL|S_NAME|N_NAME|P_PARTKEY|P_MFGR|S_ADDRESS|S_PHONE|S_COMMENT|
=========================================================================
|9938.53|Supplier#000005359       |UNITED KINGDOM           |185358.00|Manufacturer#4           |QKuHYh,vZGiwu2FWEJoLDx04|33-
429-790-6131|uriously regular requests hag|
|9937.84|Supplier#000005969                 |ROMANIA                                                  |108438.00|Manufacturer#1
|ANDENSOSmk,miq23Xfb5RWt6dvUcvt6Qa|29-520-692-3537|efully express instructions. regular requests against the slyly fin|
|9936.22|Supplier#000005250       |UNITED KINGDOM           |249.00|Manufacturer#4           |B3rqp0xbSEim4Mpy2RH J|33-320-
228-2957|etect about the furiously final accounts. slyly ironic pinto beans sleep inside the furiously|
```

```
|9923.77|Supplier#000002324        |GERMANY                    |29821.00|Manufacturer#4             |y3OD9UywSTOk|17-779-299-
1839|ackages boost blithely. blithely regular deposits c|
|9871.22|Supplier#000006373        |GERMANY                    |43868.00|Manufacturer#5             |J8fcXWsTqM|17-813-485-
8637|etect blithely bold asymptotes. fluffily ironic platelets wake furiously; blit|
---------- rows deleted -------------
|7912.91|Supplier#000004211                 |GERMANY                                                    |184210.00|Manufacturer#4
|2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG|17-266-947-7315|ay furiously regular platelets. cou|
|7894.56|Supplier#000007981        |GERMANY                    |85472.00|Manufacturer#4             |NSJ96vMROAbeXP|17-963-404-
3760|ic platelets affix after the furiously|
|7887.08|Supplier#000009792        |GERMANY                    |164759.00|Manufacturer#3            |Y28ITVeYriT3kIGdV2K8fSZ
V2UqT5H1Otz|17-988-938-4296|ckly around the carefully fluffy theodolites. slyly ironic pack|
|7871.50|Supplier#000007206        |RUSSIA                     |104695.00|Manufacturer#1            |3w fNCnrVmvJjE95sgWZzvW|32-
432-452-7731|ironic requests. furiously final theodolites cajole. final, express packages sleep. quickly reg|
|7852.45|Supplier#000005864        |RUSSIA                     |8363.00|Manufacturer#4             |WCNfBPZeSXh3h,c|32-454-883-
3821|usly unusual pinto beans. brave ideas sleep carefully quickly ironi|
|7850.66|Supplier#000001518        |UNITED KINGDOM             |86501.00|Manufacturer#1            |ONda3YJiHKJOC|33-730-383-
3892|ifts haggle fluffily pending pai|
|7843.52|Supplier#000006683                 |FRANCE                                                     |11680.00|Manufacturer#4
|2Z0JGkiv01Y00oCFwUGfviIbhzCdy|16-464-517-8943| express, final pinto beans x-ray slyly asymptotes. unusual, unusual|

resultset consists of 100 rows in 8 columns


******************
* TPC-H Query 3 0 *
******************
-- Minor modification - result set limit ( 2.1.2.9.3 )
select
          l_orderkey,
          sum(l_extendedprice * (1 - l_discount)) as revenue,
          o_orderdate,
          o_shippriority
from
          customer,
          orders,
          lineitem
where
          c_mktsegment = 'BUILDING'
          and c_custkey = o_custkey
          and l_orderkey = o_orderkey
          and o_orderdate < date '1995-03-15'
          and l_shipdate > date '1995-03-15'
group by
          l_orderkey,
          o_orderdate,
          o_shippriority
order by
          revenue desc,
          o_orderdate
LIMIT 10
;

|L_ORDERKEY|REVENUE|O_ORDERDATE|O_SHIPPRIORITY|
=============================================
|2456423.00|406181.01|1995-03-05|0.00|
|3459808.00|405838.70|1995-03-04|0.00|
|492164.00|390324.06|1995-02-19|0.00|
|1188320.00|384537.94|1995-03-09|0.00|
|2435712.00|378673.06|1995-02-26|0.00|
|4878020.00|378376.80|1995-03-12|0.00|
|5521732.00|375153.92|1995-03-13|0.00|
|2628192.00|373133.31|1995-02-22|0.00|
|993600.00|371407.46|1995-03-05|0.00|
|2300070.00|367371.15|1995-03-13|0.00|

resultset consists of 10 rows in 4 columns


******************
* TPC-H Query 4 0 *
******************
-- Minor modification - date arithmetic for month and years ( 2.2.3.3 c) )
select
          o_orderpriority,
          count(*) as order_count
from
          orders
where
          o_orderdate >= date '1993-07-01'
          and o_orderdate < add_months( date '1993-07-01' , 3 )
          and exists (
                    select    *
                    from      lineitem
                    where     l_orderkey = o_orderkey
                              and l_commitdate < l_receiptdate
          )
group by
          o_orderpriority
order by
          o_orderpriority;

|O_ORDERPRIORITY|ORDER_COUNT|
==============================
|1-URGENT       |10594.00|
|2-HIGH         |10476.00|
|3-MEDIUM       |10410.00|
|4-NOT SPECIFIED|10556.00|
|5-LOW          |10487.00|

resultset consists of 5 rows in 2 columns
```

```
******************
* TPC-H Query 5 0 *
******************
-- Minor modification - date arithmetic for month and years ( 2.2.3.3 c) )
select
          n_name,
          sum(l_extendedprice * (1 - l_discount)) as revenue
from
          customer,
          orders,
          lineitem,
          supplier,
          nation,
          region
where
          c_custkey = o_custkey
          and l_orderkey = o_orderkey
          and l_suppkey = s_suppkey
          and c_nationkey = s_nationkey
          and s_nationkey = n_nationkey
          and n_regionkey = r_regionkey
          and r_name = 'ASIA'
          and o_orderdate >= date '1994-01-01'
          and o_orderdate < add_months( date '1994-01-01' , 12)
group by
          n_name
order by
          revenue desc;

|N_NAME|REVENUE|
================
|INDONESIA                |55502041.17|
|VIETNAM                  |55295087.00|
|CHINA                    |53724494.26|
|INDIA                    |52035512.00|
|JAPAN                    |45410175.70|

resultset consists of 5 rows in 2 columns


******************
* TPC-H Query 6 0 *
******************
-- Minor modification - date arithmetic for month and years ( 2.2.3.3 c) )
select
          sum(l_extendedprice * l_discount) as revenue
from
          lineitem
where
          l_shipdate >= date '1994-01-01'
          and l_shipdate < add_months( date '1994-01-01' , 12)
          and l_discount between .06 - 0.01 and .06 + 0.01
          and l_quantity < 24;

|REVENUE|
=========
|123141078.23|

resultset consists of 1 rows in 1 columns


******************
* TPC-H Query 7 0 *
******************
select
          supp_nation,
          cust_nation,
          l_year,
          sum(volume) as revenue
from
          (
                    select
                              n1.n_name as supp_nation,
                              n2.n_name as cust_nation,
                              extract(year from l_shipdate) as l_year,
                              l_extendedprice * (1 - l_discount) as volume
                    from
                              supplier,
                              lineitem,
                              orders,
                              customer,
                              nation n1,
                              nation n2
                    where
                              s_suppkey = l_suppkey
                              and o_orderkey = l_orderkey
                              and c_custkey = o_custkey
                              and s_nationkey = n1.n_nationkey
                              and c_nationkey = n2.n_nationkey
                              and (
                                        (n1.n_name = 'FRANCE' and n2.n_name = 'GERMANY')
                                        or (n1.n_name = 'GERMANY' and n2.n_name = 'FRANCE')
                              )
                              and l_shipdate between date '1995-01-01' and date '1996-12-31'
          ) as shipping
group by
          supp_nation,
          cust_nation,
          l_year
order by
```

```
        supp_nation,
        cust_nation,
        l_year;

|SUPP_NATION|CUST_NATION|L_YEAR|REVENUE|
=======================================
|FRANCE                  |GERMANY                 |1995.00|54639732.73|
|FRANCE                  |GERMANY                 |1996.00|54633083.31|
|GERMANY                 |FRANCE                  |1995.00|52531746.67|
|GERMANY                 |FRANCE                  |1996.00|52520549.02|

resultset consists of 4 rows in 4 columns


******************
* TPC-H Query 8 0 *
******************
select
        o_year,
        sum(case
                when nation = 'BRAZIL' then volume
                else 0
        end) / sum(volume) as mkt_share
from
        (
                select
                        extract(year from o_orderdate) as o_year,
                        l_extendedprice * (1 - l_discount) as volume,
                        n2.n_name as nation
                from
                        part,
                        supplier,
                        lineitem,
                        orders,
                        customer,
                        nation n1,
                        nation n2,
                        region
                where
                        p_partkey = l_partkey
                        and s_suppkey = l_suppkey
                        and l_orderkey = o_orderkey
                        and o_custkey = c_custkey
                        and c_nationkey = n1.n_nationkey
                        and n1.n_regionkey = r_regionkey
                        and r_name = 'AMERICA'
                        and s_nationkey = n2.n_nationkey
                        and o_orderdate between date '1995-01-01' and date '1996-12-31'
                        and p_type = 'ECONOMY ANODIZED STEEL'
        ) as all_nations
group by
        o_year
order by
        o_year;

|O_YEAR|MKT_SHARE|
==================
|1995.00|0.03|
|1996.00|0.04|

resultset consists of 2 rows in 2 columns


******************
* TPC-H Query 9 0 *
******************
select
        nation,
        o_year,
        sum(amount) as sum_profit
from
        (
                select
                        n_name as nation,
                        extract(year from o_orderdate) as o_year,
                        l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as amount
                from
                        part,
                        supplier,
                        lineitem,
                        partsupp,
                        orders,
                        nation
                where
                        s_suppkey = l_suppkey
                        and ps_suppkey = l_suppkey
                        and ps_partkey = l_partkey
                        and p_partkey = l_partkey
                        and o_orderkey = l_orderkey
                        and s_nationkey = n_nationkey
                        and p_name like '%green%'
        ) as profit
group by
        nation,
        o_year
order by
        nation,
        o_year desc;

|NATION|O_YEAR|SUM_PROFIT|
```

```
==========================
|ALGERIA                  |1998.00|31342867.23|
|ALGERIA                  |1997.00|57138193.02|
|ALGERIA                  |1996.00|56140140.13|
|ALGERIA                  |1995.00|53051469.65|
|ALGERIA                  |1994.00|53867582.13|
|ALGERIA                  |1993.00|54942718.13|
---------- rows deleted --------------
|UNITED STATES            |1992.00|48671944.50|
|VIETNAM                  |1998.00|30442736.06|
|VIETNAM                  |1997.00|50309179.79|
|VIETNAM                  |1996.00|50488161.41|
|VIETNAM                  |1995.00|49658284.61|
|VIETNAM                  |1994.00|50596057.26|
|VIETNAM                  |1993.00|50953919.15|
|VIETNAM                  |1992.00|49613838.32|

resultset consists of 175 rows in 3 columns


********************
* TPC-H Query 10 0 *
********************
-- Minor modification - date arithmetic for month and years ( 2.2.3.3 c) )
-- Minor modification - result set limit ( 2.1.2.9.3 )
select
         c_custkey,
         c_name,
         sum(l_extendedprice * (1 - l_discount)) as revenue,
         c_acctbal,
         n_name,
         c_address,
         c_phone,
         c_comment
from
         customer,
         orders,
         lineitem,
         nation
where
         c_custkey = o_custkey
         and l_orderkey = o_orderkey
         and o_orderdate >= date '1993-10-01'
         and o_orderdate < add_months( date '1993-10-01' , 3)
         and l_returnflag = 'R'
         and c_nationkey = n_nationkey
group by
         c_custkey,
         c_name,
         c_acctbal,
         c_phone,
         n_name,
         c_address,
         c_comment
order by
         revenue desc
LIMIT 20
;

|C_CUSTKEY|C_NAME|REVENUE|C_ACCTBAL|N_NAME|C_ADDRESS|C_PHONE|C_COMMENT|
=====================================================================
|57040.00|Customer#000057040|734235.25|632.87|JAPAN                                |Eioyzjf4pp|22-895-641-3466|sits. slyly regular
requests sleep alongside of the regular inst|
|143347.00|Customer#000143347|721002.69|2557.47|EGYPT                           |1aReFYv,Kw4|14-742-935-3718|ggle carefully enticing
requests. final deposits use bold, bold pinto beans. ironic, idle re|
|60838.00|Customer#000060838|679127.31|2454.77|BRAZIL                            |64EaJ5vMAHWJlBOxJklpNc2RJiWE|12-913-494-9813| need
to boost against the slyly regular account|
|101998.00|Customer#000101998|637029.57|3790.89|UNITED KINGDOM                  |01c9CILnNtfOQYmZj|33-593-865-6378|ress foxes wake
slyly after the bold excuses. ironic platelets are furiously carefully bold theodolites|
|125341.00|Customer#000125341|633508.09|4983.51|GERMANY                         |S29ODD6bceU8QSuuEJznkNaK|17-582-695-5962|arefully
even depths. blithely even excuses sleep furiously. foxes use except the dependencies. ca|
|25501.00|Customer#000025501|620269.78|7725.04|ETHIOPIA                          |   W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ|15-874-808-
6793|he pending instructions wake carefully at the pinto beans. regular, final instructions along the slyly fina|
|115831.00|Customer#000115831|596423.87|5098.10|FRANCE                          |rFeBbEEyk dl ne7zV5fDrmiq1oK09wV7pxqCgIc|16-715-
386-3788|l somas sleep. furiously final deposits wake blithely regular pinto b|
|84223.00|Customer#000084223|594998.02|528.65|UNITED KINGDOM                   |nAVZCs6BaWap rrM27N 2qBnzc5WBauxbA|33-442-824-8191|
slyly final deposits haggle regular, pending dependencies. pending escapades wake |
|54289.00|Customer#000054289|585603.39|5583.02|IRAN                            |vXCxoCsU0Bad5JQI ,oobkZ|20-834-292-4707|ely special
foxes are quickly finally ironic p|
|39922.00|Customer#000039922|584878.11|7321.11|GERMANY                          |Zgy4s50l2GKN4pLDPBU8m342gIw6R|17-147-757-8036|y
final requests. furiously final foxes cajole blithely special platelets. f|
|6226.00|Customer#000006226|576783.76|2230.09|UNITED  KINGDOM                    |8gPu8,NPGkfyQQ0hcIYUGPIBWc,ybP5g,|33-657-701-
3391|ending platelets along the express deposits cajole carefully final |
|922.00|Customer#000000922|576767.53|3869.25|GERMANY                             |Az9RFaut7NkPnc5zSD2PwHgVwr4jRzq|17-945-916-
9648|luffily fluffy deposits. packages c|
|147946.00|Customer#000147946|576455.13|2030.13|ALGERIA                         |iANyZHjqhyy7Ajah0pTrYyhJ|10-886-956-3143|ithely
ironic deposits haggle blithely ironic requests. quickly regu|
|115640.00|Customer#000115640|569341.19|6436.10|ARGENTINA                       |Vtgfia9qI 7EpHgecU1X|11-411-543-4901|ost slyly
along the patterns; pinto be|
|73606.00|Customer#000073606|568656.86|1785.67|JAPAN                           |xuR0Tro5yChDfOCrjkd2ol|22-437-653-6966|he furiously
regular ideas. slowly|
|110246.00|Customer#000110246|566842.98|7763.35|VIETNAM                        |7KzflgX MDOq7sOkI|31-943-426-9837|egular deposits
serve blithely above the fl|
|142549.00|Customer#000142549|563537.24|5085.99|INDONESIA                          |ChqEoK43OysjdHbtKCp6dKqjNyvvi9|19-955-562-
2398|sleep pending courts. ironic deposits against the carefully unusual platelets cajole carefully express accounts.|
|146149.00|Customer#000146149|557254.99|1791.55|ROMANIA                         |s87fvzFQpU|29-744-164-6487| of the slyly silent
accounts. quickly final accounts across the |
|52528.00|Customer#000052528|556397.35|551.79|ARGENTINA                         |NFztyTOR10UOJ|11-208-192-3205| deposits hinder.
blithely pending asymptotes breach slyly regular re|
|23431.00|Customer#000023431|554269.54|3381.86|ROMANIA                         |HgiV0phqhaIa9aydNoIlb|29-915-458-2654|nusual, even
instructions: furiously stealthy n|
```

```
resultset consists of 20 rows in 8 columns


********************
* TPC-H Query 11 0 *
********************
-- Minor modification - Renamed keyword value to value_ ( 2.2.3.3 k) )
select
          ps_partkey,
          sum(ps_supplycost * ps_availqty) as value_
from
          partsupp,
          supplier,
          nation
where
          ps_suppkey = s_suppkey
          and s_nationkey = n_nationkey
          and n_name = 'GERMANY'
group by
          ps_partkey having
                    sum(ps_supplycost * ps_availqty) > (
                              select
                                        sum(ps_supplycost * ps_availqty) * 0.0001000000
                              from
                                        partsupp,
                                        supplier,
                                        nation
                              where
                                        ps_suppkey = s_suppkey
                                        and s_nationkey = n_nationkey
                                        and n_name = 'GERMANY'
                    )
order by
          value_ desc;

|PS_PARTKEY|VALUE_|
====================
|129760.00|17538456.86|
|166726.00|16503353.92|
|191287.00|16474801.97|
|161758.00|16101755.54|
|34452.00|15983844.72|
|139035.00|15907078.34|
|9403.00|15451755.62|
|154358.00|15212937.88|
------- rows deleted -------
|122819.00|7888881.02|
|154731.00|7888301.33|
|101674.00|7879324.60|
|51968.00|7879102.21|
|72073.00|7877736.11|
|5182.00|7874521.73|

resultset consists of 1048 rows in 2 columns


********************
* TPC-H Query 12 0 *
********************
-- Minor modification - date arithmetic for month and years ( 2.2.3.3 c) )
select
          l_shipmode,
          sum(case
                    when o_orderpriority = '1-URGENT'
                              or o_orderpriority = '2-HIGH'
                              then 1
                    else 0
          end) as high_line_count,
          sum(case
                    when o_orderpriority <> '1-URGENT'
                              and o_orderpriority <> '2-HIGH'
                              then 1
                    else 0
          end) as low_line_count
from
          orders,
          lineitem
where
          o_orderkey = l_orderkey
          and l_shipmode in ('MAIL', 'SHIP')
          and l_commitdate < l_receiptdate
          and l_shipdate < l_commitdate
          and l_receiptdate >= date '1994-01-01'
          and l_receiptdate < add_months( date '1994-01-01' , 12)
group by
          l_shipmode
order by
          l_shipmode;

|L_SHIPMODE|HIGH_LINE_COUNT|LOW_LINE_COUNT|
==============================================
|MAIL      |6202.00|9324.00|
|SHIP      |6200.00|9262.00|

resultset consists of 2 rows in 3 columns


********************
* TPC-H Query 13 0 *
********************
```

```
-- Minor modification - Outer join syntax ( 2.2.3.3 q) )
-- Minor modification - Naming of the columns of the sub select - different syntax for select-list AS clause ( 2.2.3.3 b) )
select
          c_count,
          count(*) as custdist
from
          (
                    select    c_custkey,
                              count(o_orderkey) c_count
                    from      customer, orders
                    where c_custkey = o_custkey(+)
                                        and o_comment(+) not like '%special%requests%'
                    group by
                              c_custkey
          ) as c_orders
group by
          c_count
order by
          custdist desc,
          c_count desc;

|C_COUNT|CUSTDIST|
==================
|0.00|50005.00|
|9.00|6641.00|
|10.00|6532.00|
|11.00|6014.00|
|8.00|5937.00|
|12.00|5639.00|
|13.00|5024.00|
|19.00|4793.00|
|7.00|4687.00|
|17.00|4587.00|
|18.00|4529.00|
|20.00|4516.00|
|15.00|4505.00|
|14.00|4446.00|
|16.00|4273.00|
|21.00|4190.00|
|22.00|3623.00|
|6.00|3265.00|
|23.00|3225.00|
|24.00|2742.00|
|25.00|2086.00|
|5.00|1948.00|
|26.00|1612.00|
|27.00|1179.00|
|4.00|1007.00|
|28.00|893.00|
|29.00|593.00|
|3.00|415.00|
|30.00|376.00|
|31.00|226.00|
|32.00|148.00|
|2.00|134.00|
|33.00|75.00|
|34.00|50.00|
|35.00|37.00|
|1.00|17.00|
|36.00|14.00|
|38.00|5.00|
|37.00|5.00|
|40.00|4.00|
|41.00|2.00|
|39.00|1.00|

resultset consists of 42 rows in 2 columns


********************
* TPC-H Query 14 0 *
********************
-- Minor modification - date arithmetic for month and years ( 2.2.3.3 c) )
select
          100.00 * sum(case
                    when p_type like 'PROMO%'
                              then l_extendedprice * (1 - l_discount)
                    else 0
          end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
          lineitem,
          part
where
          l_partkey = p_partkey
          and l_shipdate >= date '1995-09-01'
          and l_shipdate < add_months( date '1995-09-01', 1);

|PROMO_REVENUE|
===============
|16.38|

resultset consists of 1 rows in 1 columns


********************
* TPC-H Query 15 0 *
********************
-- Minor modification - date arithmetic for month and years ( 2.2.3.3 c) )
create view revenue0 (supplier_no, total_revenue) as
          select
                    l_suppkey,
```

```
                sum(l_extendedprice * (1 - l_discount))
        from
                lineitem
        where
                l_shipdate >= date '1996-01-01'
                and l_shipdate < add_months(date '1996-01-01' , 3)
        group by
                l_suppkey;

select
        s_suppkey,
        s_name,
        s_address,
        s_phone,
        total_revenue
from
        supplier,
        revenue0
where
        s_suppkey = supplier_no
        and total_revenue = (
                select    max(total_revenue)
                from      revenue0
        )
order by
        s_suppkey;

|S_SUPPKEY|S_NAME|S_ADDRESS|S_PHONE|TOTAL_REVENUE|
=================================================
|8449.00|Supplier#000008449       |Wp34zim9qYFbVctdW|20-469-856-8873|1772627.21|

resultset consists of 1 rows in 5 columns

drop view revenue0;

********************
* TPC-H Query 16 0 *
********************
select
        p_brand,
        p_type,
        p_size,
        count(distinct ps_suppkey) as supplier_cnt
from
        partsupp,
        part
where
        p_partkey = ps_partkey
        and p_brand <> 'Brand#45'
        and p_type not like 'MEDIUM POLISHED%'
        and p_size in (49, 14, 23, 45, 19, 3, 36, 9)
        and ps_suppkey not in (
                select    s_suppkey
                from      supplier
                where     s_comment like '%Customer%Complaints%'
        )
group by
        p_brand,
        p_type,
        p_size
order by
        supplier_cnt desc,
        p_brand,
        p_type,
        p_size;

|P_BRAND|P_TYPE|P_SIZE|SUPPLIER_CNT|
====================================
|Brand#41  |MEDIUM BRUSHED TIN|3.00|28.00|
|Brand#54  |STANDARD BRUSHED COPPER|14.00|27.00|
|Brand#11  |STANDARD BRUSHED TIN|23.00|24.00|
|Brand#11  |STANDARD BURNISHED BRASS|36.00|24.00|
|Brand#15  |MEDIUM ANODIZED NICKEL|3.00|24.00|
|Brand#15  |SMALL ANODIZED BRASS|45.00|24.00|
|Brand#15  |SMALL BURNISHED NICKEL|19.00|24.00|
----- rows deleted ------
|Brand#51  |SMALL PLATED BRASS|23.00|3.00|
|Brand#52  |MEDIUM BRUSHED BRASS|45.00|3.00|
|Brand#53  |MEDIUM BRUSHED TIN|45.00|3.00|
|Brand#54  |ECONOMY POLISHED BRASS|9.00|3.00|
|Brand#55  |PROMO PLATED BRASS|19.00|3.00|
|Brand#55  |STANDARD PLATED TIN|49.00|3.00|

resultset consists of 18314 rows in 4 columns


********************
* TPC-H Query 17 0 *
********************
select
        sum(l_extendedprice) / 7.0 as avg_yearly
from
        lineitem,
        part
where
        p_partkey = l_partkey
        and p_brand = 'Brand#23'
        and p_container = 'MED BOX'
        and l_quantity < (
                select    0.2 * avg(l_quantity)
                from      lineitem
```

```
                where     l_partkey = p_partkey
        );

|AVG_YEARLY|
============
|348406.05|

resultset consists of 1 rows in 1 columns


********************
* TPC-H Query 18 0 *
********************
-- Minor modification - result set limit ( 2.1.2.9.3 )
select
        c_name,
        c_custkey,
        o_orderkey,
        o_orderdate,
        o_totalprice,
        sum(l_quantity)
from
        customer,
        orders,
        lineitem
where
        o_orderkey in (
                select    l_orderkey
                from      lineitem
                group by  l_orderkey having sum(l_quantity) > 300
        )
        and c_custkey = o_custkey
        and o_orderkey = l_orderkey
group by
        c_name,
        c_custkey,
        o_orderkey,
        o_orderdate,
        o_totalprice
order by
        o_totalprice desc,
        o_orderdate
LIMIT 100
;

|C_NAME|C_CUSTKEY|O_ORDERKEY|O_ORDERDATE|O_TOTALPRICE|SUM(LINEITEM.L_QUANTITY)|
==========================================================================
|Customer#000128120|128120.00|4722021.00|1994-04-07|544089.09|323.00|
|Customer#000144617|144617.00|3043270.00|1997-02-12|530604.44|317.00|
|Customer#000013940|13940.00|2232932.00|1997-04-13|522720.61|304.00|
|Customer#000066790|66790.00|2199712.00|1996-09-30|515531.82|327.00|
|Customer#000046435|46435.00|4745607.00|1997-07-03|508047.99|309.00|
|Customer#000015272|15272.00|3883783.00|1993-07-28|500241.33|302.00|
|Customer#000146608|146608.00|3342468.00|1994-06-12|499794.58|303.00|
---------- rows deleted -------------
|Customer#000149842|149842.00|5156581.00|1994-05-30|411329.35|302.00|
|Customer#000010129|10129.00|5849444.00|1994-03-21|409129.85|309.00|
|Customer#000069904|69904.00|1742403.00|1996-10-19|408513.00|305.00|
|Customer#000017746|17746.00|6882.00|1997-04-09|408446.93|303.00|
|Customer#000013072|13072.00|1481925.00|1998-03-15|399195.47|301.00|
|Customer#000082441|82441.00|857959.00|1994-02-07|382579.74|305.00|
|Customer#000088703|88703.00|2995076.00|1994-01-30|363812.12|302.00|

resultset consists of 57 rows in 6 columns


********************
* TPC-H Query 19 0 *
********************
select
        sum(l_extendedprice* (1 - l_discount)) as revenue
from
        lineitem,
        part
where     (
                p_partkey = l_partkey
                and p_brand = 'Brand#12'
                and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
                and l_quantity >= 1 and l_quantity <= 1 + 10
                and p_size between 1 and 5
                and l_shipmode in ('AIR', 'AIR REG')
                and l_shipinstruct = 'DELIVER IN PERSON'
        )
        or
        (
                p_partkey = l_partkey
                and p_brand = 'Brand#23'
                and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
                and l_quantity >= 10 and l_quantity <= 10 + 10
                and p_size between 1 and 10
                and l_shipmode in ('AIR', 'AIR REG')
                and l_shipinstruct = 'DELIVER IN PERSON'
        )
        or
        (
                p_partkey = l_partkey
                and p_brand = 'Brand#34'
                and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
                and l_quantity >= 20 and l_quantity <= 20 + 10
                and p_size between 1 and 15
                and l_shipmode in ('AIR', 'AIR REG')
```

```
                and l_shipinstruct = 'DELIVER IN PERSON'
        );

|REVENUE|
=========
|3083843.06|

resultset consists of 1 rows in 1 columns


*******************
* TPC-H Query 20 0 *
*******************
-- Minor modification - date arithmetic for month and years ( 2.2.3.3 c) )
select
        s_name,
        s_address
from
        supplier,
        nation
where
        s_suppkey in (
                select    ps_suppkey
                from      partsupp
                where     ps_partkey in (
                                select    p_partkey
                                from      part
                                where     p_name like 'forest%'
                        )
                        and ps_availqty > (
                                select    0.5 * sum(l_quantity)
                                from      lineitem
                                where     l_partkey = ps_partkey
                                          and l_suppkey = ps_suppkey
                                          and l_shipdate >= date '1994-01-01'
                                          and l_shipdate < add_months( date '1994-01-01' , 12)
                        )
        )
        and s_nationkey = n_nationkey
        and n_name = 'CANADA'
order by
        s_name;

|S_NAME|S_ADDRESS|
==================
|Supplier#000000020       |iybAE,RmTymrZVYaFZva2SH,j|
|Supplier#000000091       |YV45D7TkfdQanOOZ7q9QxkyGUapU1oOWU6q3|
|Supplier#000000197       |YC2Acon6kjY3zj3Fbxs2k4Vdf7X0cd2F|
|Supplier#000000226       |83qOdU2EYRdPQAQhEtn GRZEd|
|Supplier#000000285       |Br7e1nnt1yxrw6ImgpJ7YdhFDjuBf|
|Supplier#000000378       |FfbhyCxWvcPrO8ltp9|
|Supplier#000000402       |i9Sw4DoyMhzhKXCH9By,AYSgmD|
---------- rows deleted -------------
|Supplier#000009862       |rJzweWeN58|
|Supplier#000009868       |ROjGgx5gvtkmnUUoeyy7v|
|Supplier#000009869       |ucLqxzrpBTRMewGSM29t0rNTM30g1Tu3Xgg3mKag|
|Supplier#000009899       |7XdpAHrzr1t,UQFZE|
|Supplier#000009974       |7wJ,J5DKcxSU4Kp1cQLpbcAvB5AsvKT|

resultset consists of 204 rows in 2 columns


*******************
* TPC-H Query 21 0 *
*******************
-- Minor modification - result set limit ( 2.1.2.9.3 )
select
        s_name,
        count(*) as numwait
from
        supplier,
        lineitem l1,
        orders,
        nation
where
        s_suppkey = l1.l_suppkey
        and o_orderkey = l1.l_orderkey
        and o_orderstatus = 'F'
        and l1.l_receiptdate > l1.l_commitdate
        and exists (
                select    *
                from      lineitem l2
                where     l2.l_orderkey = l1.l_orderkey
                          and l2.l_suppkey <> l1.l_suppkey
        )
        and not exists (
                select    *
                from      lineitem l3
                where     l3.l_orderkey = l1.l_orderkey
                          and l3.l_suppkey <> l1.l_suppkey
                          and l3.l_receiptdate > l3.l_commitdate
        )
        and s_nationkey = n_nationkey
        and n_name = 'SAUDI ARABIA'
group by
        s_name
order by
        numwait desc,
        s_name
LIMIT 100
;
```

```
|S_NAME|NUMWAIT|
================
|Supplier#000002829       |20.00|
|Supplier#000005808       |18.00|
|Supplier#000000262       |17.00|
|Supplier#000000496       |17.00|
|Supplier#000002160       |17.00|
|Supplier#000002301       |17.00|
|Supplier#000002540       |17.00|
---------- rows deleted -------------
|Supplier#000001337       |12.00|
|Supplier#000001916       |12.00|
|Supplier#000001925       |12.00|
|Supplier#000002039       |12.00|
|Supplier#000002357       |12.00|
|Supplier#000002483       |12.00|

resultset consists of 100 rows in 2 columns


********************
* TPC-H Query 22 0 *
********************
-- Minor modification - use of substring instead of substr ( 2.2.3.3 p) )
select
          cntrycode,
          count(*) as numcust,
          sum(c_acctbal) as totacctbal
from
          (
                    Select    substring(c_phone from 1 for 2) as cntrycode,
                              c_acctbal
                    from      customer
                    where     substring(c_phone from 1 for 2) in
                                        ('13', '31', '23', '29', '30', '18', '17')
                              and c_acctbal > (
                                        select    avg(c_acctbal)
                                        from      customer
                                        where     c_acctbal > 0.00
                                                  and substring(c_phone from 1 for 2) in
                                                            ('13', '31', '23', '29', '30', '18', '17')
                              )
                              and not exists (
                                        select    *
                                        from      orders
                                        where     o_custkey = c_custkey
                              )
          ) as custsale
group by  cntrycode
order by  cntrycode;

|CNTRYCODE|NUMCUST|TOTACCTBAL|
==============================
|13|888.00|6737713.99|
|17|861.00|6460573.72|
|18|964.00|7236687.40|
|23|892.00|6701457.95|
|29|948.00|7158866.63|
|30|909.00|6808436.13|
|31|922.00|6806670.18|

resultset consists of 7 rows in 3 columns
```

Appendix E: Seed and Input Parameters

```
# ----------------------------
# Stream 0 Seed 331130715
# ----------------------------
14        1995-01-01
2         30        BRASS     AMERICA
9         tomato
20        cyan      1994-01-01          VIETNAM
6         1995-01-01          0.02      25
17        Brand#53  WRAP PACK
18        314
8         KENYA     AFRICA    PROMO POLISHED TIN
21        GERMANY
13        unusual   requests
3         AUTOMOBILE          1995-03-08
22        14        26        21        10        31
          22        27
16        Brand#35  LARGE PLATED        16        8
          29        32        41        24        4
          12
4         1994-04-01
11        MOZAMBIQUE          0.0000003333
15        1996-01-01
1         111
10        1993-04-01
19        Brand#52  Brand#25  Brand#43  9         10
          25
5         MIDDLE EAST         1995-01-01
7         BRAZIL    KENYA
12        AIR       MAIL      1995-01-01

# ----------------------------
# Stream 1 Seed 331130716
# ----------------------------
21        UNITED STATES
3         FURNITURE 1995-03-25
18        315
5         AFRICA    1995-01-01
11        EGYPT     0.0000003333
7         ROMANIA   FRANCE
6         1995-01-01          0.08      24
20        orchid    1997-01-01          IRAQ
17        Brand#55  WRAP CAN
12        REG AIR   MAIL      1995-01-01
16        Brand#15  PROMO POLISHED      49        33
          25        22        35        16        8
          19
15        1994-01-01
13        unusual   requests
10        1994-02-01
2         18        NICKEL    EUROPE
8         FRANCE    EUROPE    PROMO BURNISHED TIN
14        1995-01-01
19        Brand#54  Brand#53  Brand#42  5         11
          22
9         smoke
22        21        31        17        16        15
          20        32
1         119
4         1996-11-01

# ----------------------------
# Stream 2 Seed 331130717
# ----------------------------
6         1995-01-01          0.05      25
17        Brand#52  SM BOX
14        1995-01-01
16        Brand#45  MEDIUM ANODIZED     2         3
          20        33        13        9         35
          11
19        Brand#11  Brand#41  Brand#41  10        12
          29
10        1994-11-01
9         salmon
2         6         TIN       AMERICA
15        1997-01-01
8         UNITED KINGDOM      EUROPE    ECONOMY BRUSHED
NICKEL
5         AMERICA   1995-01-01
22        11        23        27        33        15
          22        17
12        SHIP      FOB       1995-01-01
7         IRAQ      UNITED KINGDOM
13        unusual   accounts
18        313
1         66
4         1994-08-01
20        bisque    1996-01-01          ARGENTINA
3         AUTOMOBILE          1995-03-10
11        PERU      0.0000003333
21        MOZAMBIQUE

# ----------------------------
# Stream 3 Seed 331130718
# ----------------------------
8         KENYA     AFRICA    ECONOMY PLATED NICKEL
5         ASIA      1995-01-01
4         1997-03-01
6         1995-01-01          0.02      25
17        Brand#54  SM PACK
7         CANADA    KENYA
1         74
18        314
22        26        23        25        19        17
          24        28
14        1996-01-01
9         purple
10        1993-08-01
15        1994-01-01
11        ETHIOPIA  0.0000003333
20        lemon     1994-01-01          MOROCCO
2         44        COPPER    MIDDLE EAST
21        INDIA
19        Brand#13  Brand#24  Brand#31  5         13
          25
13        express   accounts
16        Brand#35  ECONOMY BURNISHED   10        20
          18        32        27        30        13        4
12        MAIL      FOB       1995-01-01
3         FURNITURE 1995-03-27


# ----------------------------
# Stream 4 Seed 331130719
# ----------------------------
5         MIDDLE EAST         1995-01-01
21        ALGERIA
14        1996-01-01
19        Brand#15  Brand#12  Brand#35  10        14
          21
15        1997-01-01
17        Brand#51  SM CAN
12        TRUCK     FOB       1996-01-01
6         1995-01-01          0.08      24
4         1994-12-01
9         peach
8         FRANCE    EUROPE    ECONOMY ANODIZED NICKEL
16        Brand#15  STANDARD POLISHED   9         33
          45        19        26        24        44        1
11        CHINA     0.0000003333
2         31        BRASS     AMERICA
10        1994-05-01
18        312
1         82
13        express   accounts
7         SAUDI ARABIA        FRANCE
22        26        25        14        18        19
          34        10
3         MACHINERY 1995-03-12
20        spring    1993-01-01          ETHIOPIA


# ----------------------------
# Stream 5 Seed 331130720
# ----------------------------
21        CHINA
15        1995-01-01
4         1997-07-01
6         1996-01-01          0.05      25
7         IRAQ      UNITED KINGDOM
16        Brand#45  MEDIUM BRUSHED      11        1
          18        41        5         35        26        3
19        Brand#22  Brand#45  Brand#34  6         15
          28
18        313
14        1996-01-01
22        17        31        12        22        13
          10        23
11        FRANCE    0.0000003333
13        express   accounts
3         FURNITURE 1995-03-29
1         90
2         19        NICKEL    MIDDLE EAST
5         AFRICA    1996-01-01
8         UNITED KINGDOM      EUROPE    LARGE POLISHED NICKEL
20        forest    1996-01-01          SAUDI ARABIA
12        RAIL      FOB       1996-01-01
17        Brand#53  LG BOX
10        1993-02-01
9         navy
```

```
# ----------------------------
# Stream 6 Seed 331130721
# ----------------------------
10        1993-12-01
3         MACHINERY 1995-03-14
15        1997-01-01
13        express   deposits
6         1996-01-01          0.03      25
8         MOROCCO   AFRICA    LARGE BURNISHED NICKEL
9         metallic
7         CANADA    MOROCCO
4         1995-04-01
11        ROMANIA   0.0000003333
22        15        18        14        22        24
          21        34
18        315
12        AIR       SHIP      1996-01-01
1         98
5         AMERICA   1996-01-01
16        Brand#35  PROMO BURNISHED     20        14
          21        3         11        5         37        1
2         7         TIN       ASIA
14        1996-01-01
19        Brand#24  Brand#33  Brand#23  1         16
          25
20        puff      1995-01-01          IRAN
17        Brand#55  LG PACK
21        IRAN


# ----------------------------
# Stream 7 Seed 331130722
# ----------------------------
18        313
8         GERMANY   EUROPE    MEDIUM BRUSHED BRASS
20        chartreuse          1993-01-01          UNITED
STATES
21        BRAZIL
2         45        COPPER    MIDDLE EAST
4         1997-10-01
22        14        22        15        24        11
          19        10
17        Brand#52  LG CAN
1         106
11        GERMANY   0.0000003333
9         light
19        Brand#21  Brand#15  Brand#22  6         17
          21
3         BUILDING  1995-03-31
13        express   deposits
5         ASIA      1996-01-01
7         SAUDI ARABIA        GERMANY
10        1994-09-01
16        Brand#15  SMALL PLATED        10        8
          26        50        14        9         29
          23
6         1996-01-01          0.08      24
14        1997-01-01
15        1995-01-01
12        REG AIR   SHIP      1997-01-01


# ----------------------------
# Stream 8 Seed 331130723
# ----------------------------
19        Brand#33  Brand#43  Brand#22  1         18
          28
1         114
15        1997-01-01
17        Brand#54  MED BOX
5         EUROPE    1996-01-01
8         UNITED STATES       AMERICA   MEDIUM PLATED BRASS
9         ivory
12        FOB       SHIP      1997-01-01
14        1997-01-01
7         JAPAN     UNITED STATES
4         1995-07-01
3         HOUSEHOLD 1995-03-16
20        midnight  1996-01-01          KENYA
16        Brand#45  LARGE BRUSHED       22        29
          23        2         16        12        34
          39
6         1996-01-01          0.06      25
22        24        15        20        11        27
          14        30
10        1993-06-01
13        express   deposits
2         33        STEEL     ASIA
21        ROMANIA
18        314
11        SAUDI ARABIA        0.0000003333
```

```
# ----------------------------
# Stream 9 Seed 331130724
# ----------------------------
8         MOZAMBIQUE          AFRICA    MEDIUM ANODIZED BRASS
13        express   deposits
2         20        NICKEL    AFRICA
20        white     1995-01-01          EGYPT
17        Brand#51  MED PACK
3         BUILDING  1995-03-02
6         1997-01-01          0.03      25
21        IRAQ
18        312
11        INDIA     0.0000003333
19        Brand#35  Brand#31  Brand#11  7         19
          24
10        1994-03-01
15        1995-01-01
4         1993-04-01
22        24        23        13        25        31
          17        20
1         61
7         EGYPT     MOZAMBIQUE
12        MAIL      SHIP      1997-01-01
9         goldenrod
14        1997-01-01
5         MIDDLE EAST         1997-01-01
16        Brand#35  STANDARD ANODIZED    29        50
          27        39        7         9         31
          43


# ----------------------------
# Stream 10 Seed 331130725
# ----------------------------
6         1997-01-01          0.08      24
15        1993-01-01
18        313
17        Brand#53  MED CAN
12        TRUCK     REG AIR   1997-01-01
1         69
7         VIETNAM   INDIA
2         8         TIN       ASIA
22        27        20        14        18        12
          11        23
13        special   deposits
21        CANADA
10        1994-12-01
14        1997-01-01
9         firebrick
3         HOUSEHOLD 1995-03-18
16        Brand#15  MEDIUM PLATED       12        20        8
          18        38        1         3         19
20        hot       1993-01-01          CHINA
19        Brand#32  Brand#14  Brand#15  2         20
          20
11        VIETNAM   0.0000003333
4         1995-11-01
8         INDIA     ASIA      SMALL POLISHED BRASS
5         AFRICA    1997-01-01


# ----------------------------
# Stream 11 Seed 331130726
# ----------------------------
15        1995-01-01
14        1993-01-01
18        315
17        Brand#54  JUMBO BOX
10        1993-10-01
20        sandy     1997-01-01          INDIA
16        Brand#55  ECONOMY POLISHED    32        41
          10        18        50        39        34        9
11        INDONESIA 0.0000003333
1         77
8         ALGERIA   AFRICA    SMALL BURNISHED BRASS
4         1993-08-01
22        18        16        29        26        33
          19        27
5         AMERICA   1997-01-01
12        RAIL      REG AIR   1993-01-01
3         AUTOMOBILE          1995-03-04
9         cyan
21        VIETNAM
2         46        COPPER    AFRICA
13        special   packages
6         1997-01-01          0.06      24
19        Brand#45  Brand#52  Brand#54  7         10
          28
7         JORDAN    ALGERIA
```

Appendix F: Benchmark programs and Scripts

```
# ------------------------------------------------------------------------------------------
# tpc_h_run_full.sh (Main-Script)
# ------------------------------------------------------------------------------------------
#!/bin/bash
# param <nodes> <exadir> <scalefactor> <streams> <instances per node> <refresh functions while
queries> <logdir>
echo $*
if [ $# -ne 7 ]
then
    echo "Usage: $0 <nodes> <exadir> <scalefactor> <streams> <instances per node> <refresh
functions while queries> <logdir>"
    echo "<refresh functions while queries> may be yes or no"
    exit 1
fi

export NODELIST="$1"
export NODES="`echo $NODELIST | tr ',' '\n' | wc -l`"
SPACE_SEP_NODES="`echo -n "$NODES" | tr ',' ' '`"
export ROOTHOST=`echo $NODELIST | sed 's/\([^,]*\),.*/\1/'`
export EXADIR="$2"
export EXABINDIR="$EXADIR/bin"
export LOADERBINDIR="$EXABINDIR"
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$EXADIR/lib
SF="$3"
STREAMS="$4"
INST_PN="$5"
RF_WHILE_Q="$6"
LOGDIR="$7/`date +'%y%m%d%H%M%S'`"
LOGFILE="$LOGDIR/test.log"
ROOTPORT=26822

export TPC_BASEDIR=$HOME/TPCH
export TPC_SCRIPTSDIR=$TPC_BASEDIR/scripts
export TPC_SINGLENODE_CSVDIR="/d02_data/csv/single"
export TPC_SQLDIR=$TPC_BASEDIR/sql
export DSS_CONFIG=$TPC_BASEDIR/tpch_20070105

### Startup of database
mkdir -p $LOGDIR
mkdir -p $LOGDIR/run1
mkdir -p $LOGDIR/run2
echo "TPC Benchmark H started" >$LOGFILE
echo "-----------------------" >>$LOGFILE
echo "SCRIPT  : $0" >> $LOGFILE
echo "ROOTHOST: $ROOTHOST" >>$LOGFILE
echo "NODES   : $NODELIST" >> $LOGFILE
echo "SCALE   : $SF" >>$LOGFILE
echo "STREAMS : $STREAMS" >> $LOGFILE
echo "INSTPN  : $INST_PN" >>$LOGFILE
echo "LOGDIR  : $LOGDIR" >>$LOGFILE
echo "SCRIPTS : $TPC_SCRIPTSDIR" >>$LOGFILE
echo "-----------------------" >>$LOGFILE

echo >> $LOGFILE
echo "Start of Test at; `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`"
>>$LOGFILE

export EXAPLUS=$EXABINDIR/Console/exaplus

###### Phase I (Load-Test)

### Initialization of database and user

$EXAPLUS -u sys -P exasol -h $ROOTHOST -p $ROOTPORT -f $TPC_SQLDIR/create_user.sql

### Initialization of database-schema

TIMESFILE=$LOGDIR/run1/alltimes.log
LOAD_START="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`"
echo -n "Load Test; $LOAD_START " > $TIMESFILE
echo "Start of Load test at ; $LOAD_START" >> $LOGFILE
$EXAPLUS -p $ROOTPORT -h $ROOTHOST -u tpcuser -P tpcuser -f $TPC_SQLDIR/create_schema.sql
>>$LOGDIR/create_schema.log 2>&1

### Load

$TPC_SCRIPTSDIR/load_init_parallel.sh $ROOTHOST $ROOTPORT tpcuser tpcuser tpc $SF $INST_PN
>>$LOGFILE 2>&1
sync

### Initialization of indices
echo "Start of database optimization at ; `date +'%m/%d/%Y %k:%M:%S.%N' | sed
's/\(\....\)....../\1/g'`" >> $LOGFILE
optimizeLineitem()
{
    cat $TPC_SQLDIR/create_indices.sql | grep -i -v -e NATION -e REGION -e " PART" -e ORDERS -e
PARTSUPP -e CUSTOMER -e SUPPLIER -e ANALYZE | $EXAPLUS -p $ROOTPORT -h $ROOTHOST -u tpcuser -P
tpcuser 1>>$LOGDIR/optimizeLineitem.log 2>&1
}

optimizeOther()
{
    cat $TPC_SQLDIR/create_indices.sql | grep -i -v -e LINEITEM -e ANALYZE | $EXAPLUS -p
$ROOTPORT -h $ROOTHOST -u tpcuser -P tpcuser 1>>$LOGFILE 2>&1
}

optimizeLineitem &
optimizeOther &
wait

cat $LOGDIR/optimizeLineitem.log >> $LOGFILE
rm -f $LOGDIR/optimizeLineitem.log

cat $TPC_SQLDIR/create_indices.sql | grep -i -v -e ENFORCE -e OPTIMIZE -e SELECT | $EXAPLUS -p
$ROOTPORT -h $ROOTHOST -u tpcuser -P tpcuser 1>>$LOGFILE 2>&1

echo "End of database optimization at ; `date +'%m/%d/%Y %k:%M:%S.%N' | sed
's/\(\....\)....../\1/g'`" >> $LOGFILE
sync

LOAD_END="`date +'%m/%d/%Y %H:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`"
echo "End of Load test at ; $LOAD_END" >> $LOGFILE
echo -n "; $LOAD_END " >> $TIMESFILE
LOAD_TIME="""`$TPC_SCRIPTSDIR/timediff "$LOAD_START" "$LOAD_END"`"""
echo "; $LOAD_TIME" >> $TIMESFILE
sync

# execute tables.sql
cat $TPC_SQLDIR/tables.sql | $EXAPLUS -p $ROOTPORT -h $ROOTHOST -u tpcuser -P tpcuser
>>$LOGDIR/tables.log 2>&1

### Create queries for the query-streams

echo "Generating substitution parameters for performance test" >> $LOGFILE
```

```
$TPC_SCRIPTSDIR/make_queries.sh $LOGDIR/run1 $SF $STREAMS "$LOAD_END" >> $LOGFILE
cp $LOGDIR/run1/stream*.sql $LOGDIR/run2

echo "Phase I finished at ; " `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'` >>$LOGFILE
sync

###### Phase II (Power-Test 1)

U=1

echo "Start of Phase II at ; " `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'` >>$LOGFILE

START="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`"
echo -n "RF1 for stream/pair ; 0 ; $U ; $START " >> $TIMESFILE
echo "Start of RF1 for stream/pair ; 0 ; $U ; $START" >> $LOGFILE
$TPC_SCRIPTSDIR/load_rf1.sh $U $ROOTHOST $ROOTPORT tpcuser tpcuser tpc $SF $INST_PN >>$LOGFILE 2>&1
END="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`"
echo "End of RF1 for stream/pair ; 0 ; $U ; $END" >> $LOGFILE
echo -n "; $END " >> $TIMESFILE
TIMEUSED="""`$TPC_SCRIPTSDIR/timediff "$START" "$END"`"""
echo "; $TIMEUSED" >> $TIMESFILE

echo "Start of querystream 0 at ; `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`" >> $LOGFILE
$TPC_SCRIPTSDIR/query_streams/querystreamex $LOGDIR/run1/stream0.sql $LOGDIR/run1/stream0.log $LOGDIR/run1/stream0.log2 0 $ROOTHOST $ROOTPORT tpcuser tpcuser >>$LOGFILE 2>&1
echo "End of querystream 0 at ; `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`" >> $LOGFILE

START="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`"
echo -n "RF2 for stream/pair ; 0 ; $U ; $START " >> $TIMESFILE
echo "Start of RF2 for stream/pair ; 0 ; $U ; $START" >>$LOGFILE
$TPC_SCRIPTSDIR/query_streams/deletestreamex $TPC_SINGLENODE_CSVDIR/Database_SF$SF/delete.1 1 tpc $ROOTHOST $ROOTPORT tpcuser tpcuser >>$LOGFILE 2>&1
END="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`"
echo "End of RF2 for stream/pair ; 0 ; $U ; $END" >>$LOGFILE
echo -n "; $END " >> $TIMESFILE
TIMEUSED="""`$TPC_SCRIPTSDIR/timediff "$START" "$END"`"""
echo "; $TIMEUSED" >> $TIMESFILE

echo "End of Phase II at ; " `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'` >>$LOGFILE
sync

###### Phase III (Throughput-Test 1)

TP_START="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`"
echo "Start of Phase III at ; $TP_START" >>$LOGFILE

for ((S=1;S<$STREAMS+1;S++))
do
        echo "Start of querystream $S at ; `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`" >> $LOGFILE
    ($TPC_SCRIPTSDIR/query_streams/querystreamex $LOGDIR/run1/stream$S.sql $LOGDIR/run1/stream$S.log $LOGDIR/run1/stream$S.log2 $S $ROOTHOST $ROOTPORT tpcuser tpcuser >>$LOGFILE 2>&1 ; echo "End of querystream $S at; `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`" >>$LOGFILE) &
done
if [ "$RF_WHILE_Q" != yes ] ; then
    wait
fi
STREAM_U=1
for ((U=2;U<=$STREAMS+1;U++))
do
        START="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`"
        echo -n "RF1 for stream/pair ; $STREAM_U ; $U ; $START " >> $TIMESFILE
        echo "Start of RF1 for stream/pair ; $STREAM_U ; $U ; $START" >> $LOGFILE
        $TPC_SCRIPTSDIR/load_rf1.sh $U $ROOTHOST $ROOTPORT tpcuser tpcuser tpc $SF $INST_PN >>$LOGFILE 2>&1
        END="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`"
        echo "End of RF1 for stream/pair ; $STREAM_U ; $U ; $END" >>$LOGFILE
        echo -n "; $END " >> $TIMESFILE
        TIMEUSED="""`$TPC_SCRIPTSDIR/timediff "$START" "$END"`"""
        echo "; $TIMEUSED" >> $TIMESFILE

        START="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`"
        echo -n "RF2 for stream/pair ; $STREAM_U ; $U ; $START " >> $TIMESFILE
        echo "Start of RF2 for stream/pair ; $STREAM_U ; $U ; $START" >> $LOGFILE
        $TPC_SCRIPTSDIR/query_streams/deletestreamex $TPC_SINGLENODE_CSVDIR/Database_SF$SF/delete.$U $U tpc $ROOTHOST $ROOTPORT tpcuser tpcuser >>$LOGFILE 2>&1
        END="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`"
        echo "End of RF2 for stream/pair ; $STREAM_U ; $U ; $END" >>$LOGFILE
        echo -n "; $END " >> $TIMESFILE
        TIMEUSED="""`$TPC_SCRIPTSDIR/timediff "$START" "$END"`"""
        echo "; $TIMEUSED" >> $TIMESFILE

        STREAM_U=$((STREAM_U+1))
done
if [ "$RF_WHILE_Q" = yes ] ; then
    wait
fi

TP_END="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`"
TPTIME="""`$TPC_SCRIPTSDIR/timediff "$TP_START" "$TP_END"`"""
echo "Throughput Test ; $TP_START ; $TP_END ; $TPTIME" >> $TIMESFILE
echo "End of Phase III at ; $TP_END" >>$LOGFILE
sync

###### Phase IV (Power-Test 2)

echo "Start of Phase IV at ; " `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'` >>$LOGFILE
TIMESFILE=$LOGDIR/run2/alltimes.log

((U=STREAMS+2))
START="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`"
echo -n "RF1 for stream/pair ; 0 ; $U ; $START " >> $TIMESFILE
echo "Start of RF1 for stream/pair ; 0 ; $U ; $START" >> $LOGFILE
$TPC_SCRIPTSDIR/load_rf1.sh $U $ROOTHOST $ROOTPORT tpcuser tpcuser tpc $SF $INST_PN >>$LOGFILE 2>&1
END="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`"
echo "End of RF1 for stream/pair ; 0 ; $U ; $END" >> $LOGFILE
echo -n "; $END " >> $TIMESFILE
TIMEUSED="""`$TPC_SCRIPTSDIR/timediff "$START" "$END"`"""
echo "; $TIMEUSED" >> $TIMESFILE

echo "Start of querystream 0 at ; `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`" >> $LOGFILE
$TPC_SCRIPTSDIR/query_streams/querystreamex $LOGDIR/run2/stream0.sql $LOGDIR/run2/stream0.log $LOGDIR/run2/stream0.log2 0 $ROOTHOST $ROOTPORT tpcuser tpcuser >>$LOGFILE 2>&1
echo "End of querystream 0 at ; `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`" >> $LOGFILE

START="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)...../\1/g'`"
echo -n "RF2 for stream/pair ; 0 ; $U ; $START " >> $TIMESFILE
echo "Start of RF2 for stream/pair ; 0 ; $U ; $START" >>$LOGFILE
```

```
$TPC_SCRIPTSDIR/query_streams/deletestreamex $TPC_SINGLENODE_CSVDIR/Database_SF$SF/delete.$U $U
tpc $ROOTHOST $ROOTPORT tpcuser tpcuser >>$LOGFILE 2>&1
END="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`"
echo "End of RF2 for stream/pair ; 0 ; $U ; $END" >>$LOGFILE
echo -n "; $END " >> $TIMESFILE
TIMEUSED="""`$TPC_SCRIPTSDIR/timediff "$START" "$END"`"""
echo "; $TIMEUSED" >> $TIMESFILE

echo "End of Phase IV at ; " `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`
>>$LOGFILE
sync

###### Phase V (Throughput-Test 2)

TP_START="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`"
echo "Start of Phase V at ; $TP_START" >>$LOGFILE

for ((S=1;S<$STREAMS+1;S++))
do
        echo "Start of querystream $S at ; `date +'%m/%d/%Y %k:%M:%S.%N' | sed
's/\(\....\)....../\1/g'`" >> $LOGFILE
    ($TPC_SCRIPTSDIR/query_streams/querystreamex                    $LOGDIR/run2/stream$S.sql
$LOGDIR/run2/stream$S.log $LOGDIR/run2/stream$S.log2 $S $ROOTHOST $ROOTPORT tpcuser tpcuser
>>$LOGFILE 2>&1 ; echo "End of querystream $S at; `date +'%m/%d/%Y %k:%M:%S.%N' | sed
's/\(\....\)....../\1/g'`" >>$LOGFILE) &
done
if [ "$RF_WHILE_Q" != yes ] ; then
    wait
fi
STREAM_U=1
for ((U=$STREAMS+3;U<=$STREAMS+$STREAMS+2;U++))
do
        START="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`"
        echo -n "RF1 for stream/pair ; $STREAM_U ; $U ; $START " >> $TIMESFILE
        echo "Start of RF1 for stream/pair ; $STREAM_U ; $U ; $START" >> $LOGFILE
        $TPC_SCRIPTSDIR/load_rf1.sh $U $ROOTHOST $ROOTPORT tpcuser tpcuser tpc $SF $INST_PN
>>$LOGFILE 2>&1
        END="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`"
        echo "End of RF1 for stream/pair ; $STREAM_U ; $U ; $END" >>$LOGFILE
        echo -n "; $END " >> $TIMESFILE
        TIMEUSED="""`$TPC_SCRIPTSDIR/timediff "$START" "$END"`"""
        echo "; $TIMEUSED" >> $TIMESFILE

        START="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`"
        echo -n "RF2 for stream/pair ; $STREAM_U ; $U ; $START " >> $TIMESFILE
        echo "Start of RF2 for stream/pair ; $STREAM_U ; $U ; $START" >> $LOGFILE
        $TPC_SCRIPTSDIR/query_streams/deletestreamex
$TPC_SINGLENODE_CSVDIR/Database_SF$SF/delete.$U $U tpc $ROOTHOST $ROOTPORT tpcuser tpcuser
>>$LOGFILE 2>&1
        END="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`"
        echo "End of RF2 for stream/pair ; $STREAM_U ; $U ; $END" >>$LOGFILE
        echo -n "; $END " >> $TIMESFILE
        TIMEUSED="""`$TPC_SCRIPTSDIR/timediff "$START" "$END"`"""
        echo "; $TIMEUSED" >> $TIMESFILE

        STREAM_U=$((STREAM_U+1))
done
if [ "$RF_WHILE_Q" = yes ] ; then
    wait
fi

TP_END="`date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`"
TPTIME="""`$TPC_SCRIPTSDIR/timediff "$TP_START" "$TP_END"`"""
echo "Throughput Test ; $TP_START ; $TP_END ; $TPTIME" >> $TIMESFILE
echo "End of Phase V at ; $TP_END" >>$LOGFILE
sync

### Extract and format important times for presentation
$TPC_SCRIPTSDIR/calculate_results.sh        $LOGDIR/run1        $STREAMS        $TPC_SCRIPTSDIR
>$LOGDIR/run1/result.csv
$TPC_SCRIPTSDIR/calculate_results.sh        $LOGDIR/run2        $STREAMS        $TPC_SCRIPTSDIR
>$LOGDIR/run2/result.csv

# execute dbcheck.sql
cat $TPC_SQLDIR/dbcheck.sql | $EXAPLUS -p $ROOTPORT -h $ROOTHOST -u tpcuser -P tpcuser
>>$LOGDIR/dbcheck.log 2>&1

# dw $ENVIRONMENT shutdown

echo "End of Test at ; " `date +'%m/%d/%Y %k:%M:%S.%N' | sed 's/\(\....\)....../\1/g'`
>>$LOGFILE



# ----------------------------------------------------------------------------------------
# make_queries.sh
# ----------------------------------------------------------------------------------------
#!/bin/bash
# params: <logdir> <scalefactor> <streamsTP> <endtime of load>
LOGDIR=$1
SF=$2
STREAMS=$3
LOADEND="$4"

QRANDOM=`echo "$LOADEND" | sed 's%\(.\+\)/\(.\+\)/....
\+\(.\+\):\(..\):\(..\)\....%\1\2\3\4\5%' | sed 's%^0%%'`

export DSS_CONFIG=$TPC_BASEDIR/tpch_20070105
cd $DSS_CONFIG
for ((U=0;U<=$STREAMS;U++))
do
  echo "QGen seed; stream $U; $QRANDOM"
  $DSS_CONFIG/qgen -s $SF -c -i $TPC_SQLDIR/initstream.sql -p $U -r $QRANDOM -l
$LOGDIR/substitution_parameters_$U.txt > $LOGDIR/stream$U.sql
  echo '# ----------------------------' >> $LOGDIR/substitution_parameters.txt
  echo "# Stream $U Seed $QRANDOM" >> $LOGDIR/substitution_parameters.txt
  echo '# ----------------------------' >> $LOGDIR/substitution_parameters.txt
  cat $LOGDIR/substitution_parameters_$U.txt >> $LOGDIR/substitution_parameters.txt
  echo >> $LOGDIR/substitution_parameters.txt
  dos2unix $LOGDIR/stream$U.sql
  ((QRANDOM=QRANDOM+1))
done



# ----------------------------------------------------------------------------------------
# initstream.sql
# ----------------------------------------------------------------------------------------
OPEN SCHEMA tpc;
COMMIT;



# ----------------------------------------------------------------------------------------
# load_rf1.sh (RF1)
# ----------------------------------------------------------------------------------------
#!/bin/bash

if [ $# -ne 8 ]; then
```

```
        echo "Syntax: $0 <num> <host> <port> <user> <password> <schema> <SF> <instances-per-
node>"
        exit 1
fi

num="$1"
host="$2"
port="$3"
user="$4"
pass="$5"
schema="$6"
SF="$7"
INST_PN="$8"
ParLoader=$LOADERBINDIR/EXAloader_PARSYNC

TPC_LOCALDIR="/d02_data/csv/local/TPC_Database_SF${SF}_${INST_PN}x_${NODES}N"

ORDERS_COLUMNS="O_ORDERKEY:DECIMAL(11)|O_CUSTKEY:DECIMAL(11)|O_ORDERSTATUS:CHAR(1)|O_TOTALPRICE
:DECIMAL(12,2)|O_ORDERDATE:DATE|O_ORDERPRIORITY:CHAR(15)|O_CLERK:CHAR(15)|O_SHIPPRIORITY:DECIMA
L(10)|O_COMMENT:VARCHAR(79)"
LINEITEM_COLUMNS="L_ORDERKEY:DECIMAL(11)|L_PARTKEY:DECIMAL(11)|L_SUPPKEY:DECIMAL(11)|L_LINENUMB
ER:DECIMAL(10)|L_QUANTITY:DECIMAL(12,2)|L_EXTENDEDPRICE:DECIMAL(12,2)|L_DISCOUNT:DECIMAL(12,2)|
L_TAX:DECIMAL(12,2)|L_RETURNFLAG:CHAR(1)|L_LINESTATUS:CHAR(1)|L_SHIPDATE:DATE|L_COMMITDATE:DATE
|L_RECEIPTDATE:DATE|L_SHIPINSTRUCT:CHAR(25)|L_SHIPMODE:CHAR(10)|L_COMMENT:VARCHAR(44)"

( $ParLoader -q -l csv2exa \
    -s $TPC_LOCALDIR/metafile.RF1.csv.u$num \
    -d $user:$schema/$pass@$host:$port -a \
    -t ORDERS -c "`eval echo \\${ORDERS_COLUMNS}`" \
    -t LINEITEM -c "`eval echo \\${LINEITEM_COLUMNS}`" \
| tr -d '\n' | sed -e 's/rows loaded\.//g' | sed -e 's/^\([0-9]*\) \([0-9]*\) $/ORDERS rows
loaded; \1\nLINEITEM rows loaded; \2\n/' \
) || { echo "RF1 loading failed (step 1 of 1)." ; exit 1 ; }

exit 0



# ------------------------------------------------------------------------------------------
# deletestream.cpp (RF2)
# ------------------------------------------------------------------------------------------
#include "querystream_for_delete.h"

#include <iostream>
using std::cout;
using std::cerr;
using std::endl;

#include <fstream>
using std::ofstream;
using std::ifstream;

#include <string>
using std::string;

#include <sstream>
using std::ostringstream;

#include <sys/time.h>

/** \brief Calculates the time passed between t1 and t2. */
double timediff( timeval t1, timeval t2 )
{
    double passed = 1.0e-6 * ( t2.tv_usec - t1.tv_usec );
    passed += ( t2.tv_sec - t1.tv_sec );
    if ( passed < 0 )
        passed = -passed;
    return passed;
}

/** \brief Deletes from a given string all occurences of another string at beginning and end.
*/
string trim( string &s, const string drop = " " )
{
    string r = s.erase( s.find_last_not_of(drop)+1 );
    return r.erase( 0, r.find_first_not_of(drop) );
}

const int KEY_COUNT_PER_STEP = 1500000;

int main( int argc, char** argv )
{
    timeval stream_starttime;
    timeval stream_endtime;
    timeval starttime;
    timeval endtime;

    bool firsttime = true;
    if ( argc!=8 && argc!=9 )
    {
        cout << "usage: " << argv[0] << " <inputfile> <streamid> <schema> <host> <port> <user>
<password> [<logfile>]" << endl;
        cout << endl;
        cout << "\tinputfile name of the file containing the queries to execute" << endl;
        cout << "\tstreamid  an identifier used for this stream" << endl;
        cout << "\thost      name of host to connect to" << endl;
        cout << "\tport      port number on host to connect to" << endl;
                cout << "\tschema     name of a schema to open prior to any statements in
inputfile (default: open no schema)" << endl;
        cout << "\tuser      name of user to connect to EXASolution (default: sys)" << endl;
        cout << "\tpassword  password used to authenticate user (default: exasol)" << endl;
        return 1;
    }
    Configuration config;
    config.host = argv[4];
    config.port = argv[5];
    config.user     = argv[6];
    config.password = argv[7];

          std::string initialSchema = argv[3];
    std::string myStreamID = argv[2];

          bool hasLog = false;
          ofstream logfile;

          if ( argc == 9 )
          {
          gettimeofday(&stream_starttime, 0);

                    logfile.open( argv[8] );
                    hasLog = true;
          }

          if ( hasLog )
          {
                    logfile.setf(std::ios::fixed);
                    logfile.precision(2);

              logfile << "==============================";
              for ( int i=0; i< myStreamID.size(); i++ )
```

```
          logfile << "=";
          logfile << '\n';
          logfile << "= Logfile for delete stream " << myStreamID << " =\n";
          logfile << "==============================";
          for ( int i=0; i< myStreamID.size(); i++ )
          logfile << "=";
          logfile << "\n" << endl;
          }

    ifstream inputfile( argv[1] );
          if ( !inputfile )
          {
                    cerr << "delete stream " << myStreamID << ": couldn't open inputfile '" <<
argv[1] << "'" << endl;
                    if ( hasLog )
                              logfile << "delete stream " << myStreamID << ": couldn't open
inputfile '" << argv[1] << "'" << endl;
                    exit( -1 );
          }

    QueryStream qs( myStreamID, hasLog ? &logfile : NULL );

          //cout << "hostname: " << config.host << endl;
          //cout << "port    : " << config.port << endl;
          //cout << "user    : " << config.user << endl;
          //cout << "password: " << config.password << endl;
    if ( ! qs.connectToServer( config ) )
    {
        qs.disconnectFromServer();
        return -1;
    }

          // open the selected schema if any was given
          if ( initialSchema != "" )
          {
                    Handle schemaHandle;
                    std::string openSchemaStatement = "open schema " + initialSchema +";";
                    qs.prepare( schemaHandle, openSchemaStatement);
                    qs.execute( schemaHandle, 0);
                    qs.done( schemaHandle );
          }
          gettimeofday(&starttime, 0 );

          uint keys[KEY_COUNT_PER_STEP];
          size_t keysRead = 0;

          // read orderkey values from file
          string statementOrders = "delete from orders where o_orderkey = ?;";
          string statementLineitem = "delete from lineitem where l_orderkey = ?;";

          Handle ordersHandle;
          Handle lineitemHandle;

          //cout << statementOrders << endl;
          qs.prepare( ordersHandle, statementOrders );

          //cout << statementLineitem << endl;
          qs.prepare( lineitemHandle, statementLineitem );

          //cout << statementOrders << endl;
          qs.bindIntParameter( ordersHandle, keys, KEY_COUNT_PER_STEP );
          //cout << statementLineitem << endl;
          qs.bindIntParameter( lineitemHandle, keys, KEY_COUNT_PER_STEP );

          while ( !inputfile.eof() )
          {
                    keysRead = 0;
                    while ( (keysRead < KEY_COUNT_PER_STEP) && (! inputfile.eof()) )
                    {
                              inputfile >> keys[keysRead];
                              if ( inputfile.eof() )
                                        break;
                              keysRead++;
                    }

                    //cout << keysRead << " keys read:" << endl;
                    //if ( hasLog )
                    //{
                    //        for ( int i=0; i<keysRead; i++ )
                    //                logfile << keys[i] << endl;
                    //}
                    //cout << statementOrders << endl;
                    qs.execute( ordersHandle, keysRead );
                    //cout << statementLineitem << endl;
                    qs.execute( lineitemHandle, keysRead );
          }

          //cout << "done" << endl;
          qs.done( ordersHandle );
          qs.done( lineitemHandle );

          //cout << statementOrders << " done" << endl;

          //cout << "commit" << endl;
          Handle commitHandle;
          string commit = "commit;";
          qs.prepare( commitHandle, commit );
          qs.execute( commitHandle, 0);
          qs.done( commitHandle );
          //cout << "commit done" << endl;

          if ( hasLog )
          {
          gettimeofday( &endtime, 0 );
          logfile << "\n----------\n";
          logfile << "- Summary -\n";
          logfile << "----------\n\n";
          logfile << "started  at: " << ctime(&(starttime.tv_sec))
                    << " (epoch: " << starttime.tv_sec << " s, " << starttime.tv_usec
                    << " usec"<< ")\n";
          logfile << "finished at: " << ctime(&(endtime.tv_sec))
                    << " (epoch: " << endtime.tv_sec << " s, " << endtime.tv_usec
                    << " usec" << ")\n";
          logfile << "time used  : " << timediff(starttime, endtime) << " s\n\n";

          logfile << "************************\n";
          logfile << "* End of delete stream *\n";
          logfile << "************************\n\n";
          }

    qs.disconnectFromServer();

          if ( hasLog )
          {
          gettimeofday( &stream_endtime, 0 );
          logfile << "--------------------------";
          for ( int i=0; i<myStreamID.size(); i++ )
          logfile << "-";
          logfile << "\n- Summary of deletestream " << myStreamID << " -\n";
          logfile << "--------------------------";
```

```
          for ( int i=0; i<myStreamID.size(); i++ )
          logfile << "-";
          logfile << "\n\nstarted  at: " << ctime(&(stream_starttime.tv_sec))
                  << " (epoch: " << stream_starttime.tv_sec << " s, "
                  << stream_starttime.tv_usec << " usec"<< ")\n";
          logfile << "finished at: " << ctime(&(stream_endtime.tv_sec))
                  << " (epoch: " << stream_endtime.tv_sec << " s, "
                  << stream_endtime.tv_usec << " usec" << ")\n";
          logfile << "time used  : " << timediff(stream_starttime, stream_endtime)
                  << " s\n\n";
          }

    return 0;
}



# ------------------------------------------------------------------------------------------
# querystream_for_delete.h
# ------------------------------------------------------------------------------------------
#ifndef QUERYSTREAM_H
#define QUERYSTREAM_H

#include <string>
#include <iostream>

#include "exaCInterface.h"

/** \brief The maximum length of a column name. */
static const size_t maxColumnNameSize = 40;

/** \brief A structure to describe connection information to a database.
  * Used in QueryStream::connectToServer().
  * \sa QueryStream::connectToServer() */
struct Configuration
{
    /** \brief The hostname of the server. */
    char *host;
    /** \brief The port of the server to connect to. */
    char *port;
    /** \brief The user name used for authentication to the database. */
    char *user;
    /** \brief The password of the user. */
    char *password;
};

struct Handle
{
          size_t max_size;
    /** \brief The only statement handle used. */
    SQLHSTMT hstmt;
          //SQLUSMALLINT *statusArray;
          SQLUSMALLINT parameterProcessed;
          SQLINTEGER *indArray;

          Handle();
          ~Handle();
};

/** \brief The class represents a query stream to a server.
  * The query stream can be used to send any query as text to the server and
  * retrieve the result. */
class QueryStream
{
public:
    /** \brief Constructor.
      * \param streamId The identifier used to identify this query stream.
      * \param logfile The stream to be used for logging. */
    QueryStream( std::string streamId, std::ostream* logfile = &std::cout );
    /** \brief Tries to establish a connection to a server.
      * \param configuration The connection and login information for the server.
      * \return true if the connection could be established, false otherwise. */
    bool connectToServer( Configuration configuration );

          bool prepare( Handle &handle, std::string statement );
          bool bindIntParameter( Handle &handle, uint *parameterArray, size_t size );
          bool execute( Handle &handle, size_t current_size );
          bool done( Handle &handle );
    /** \brief Closes the connection to the server.
      * \return true if the connection was closed successfully, false if errors occured. */
    bool disconnectFromServer();
protected:
          void setSize( Handle &handle, long long int new_size );
    /** \brief The maximum size of a string. */
    static const size_t maxStringSize  = 200;

    /** \brief The stream identifier. */
    std::string myID;

    /** \brief The environment handle used. */
    SQLHENV  henv;
    /** \brief The connection handle used. */
    SQLHDBC  hdbc;

    /** \brief The stream used for logging results.*/
    std::ostream* logfile;
};

#endif // QUERYSTREAM_H



# ------------------------------------------------------------------------------------------
# querystream_for_delete.cpp
# ------------------------------------------------------------------------------------------
#include "querystream_for_delete.h"
#include <iostream>

#ifdef DEBUG
#define DEBUGOUT( X ) cout << "DEBUG:" << x
#else
#define DEBUGOUT( X )
#endif

#define LOG *logfile

using std::cout;
using std::endl;
using std::cerr;

void errorDiagnosis( void *handle )
{
          if ( handle != 0 )
          {
                    SQLCHAR state[6];
                    SQLINTEGER nativeCode;
                    SQLCHAR message[100];
                    SQLSMALLINT messageLength;
                    SQLSMALLINT i=0;
                    int retcode;
```

```
                    cout << "doing error diagnosis" << endl;
                    while(  (  retcode  =  EXAGetDiagRec(  SQL_HANDLE_STMT,  handle,  i,  state,
&nativeCode, message, 100, &messageLength ) ) == SQL_SUCCESS )
                    {
                              cout << "message: " << message << endl;
                              i++;
                    }
          }
}

SQLRETURN execCritical(SQLRETURN retcode, void* handle = 0)
{
    switch(retcode)
    {
    case SQL_INVALID_HANDLE:
                    cerr << "invalid handle" << endl;
                    errorDiagnosis( handle );
        throw "ERROR use of invalid handle!\n";
        break;
    case SQL_ERROR:
                    cerr << "SQL error" << endl;
                    errorDiagnosis( handle );
        throw "An error occured while executing command!\n";
        break;
          case SQL_SUCCESS_WITH_INFO:
                    cerr << "Problem: Has Info, which should not be" << endl;
                    errorDiagnosis( handle );
                    throw "Problem: Has Info, which should not be!\n";
          break;
          case SQL_STILL_EXECUTING:
                    cerr << "Still executing. This should not be should only";
                    cerr << "return on successful execution!" << endl;
                    errorDiagnosis( handle );
                    throw "Unexpectedly still executing!\n";
                    break;
          case SQL_NO_DATA:
          case SQL_SUCCESS:
        break;
          default:
                    // should not reach here...
                    cerr << "unexpected return code: " << retcode << endl;
                    errorDiagnosis( handle );
                    throw "unexpected return code!\n";
                    break;
    }
    return retcode;
}

Handle::Handle()
{
          max_size = 0;
          hstmt = NULL;
          indArray = NULL;
          //statusArray = NULL;
          parameterProcessed = 0;
}

Handle::~Handle()
{
          if ( indArray != 0 )
          {
                    free( indArray );
                    indArray = 0;
          }
          /*if ( statusArray != 0 )
          {
                    free( statusArray );
                    statusArray = 0;
          }*/
}

QueryStream::QueryStream( std::string streamId, std::ostream* logfile_ )
{
    myID  = streamId;
    henv  = NULL;
    hdbc  = NULL;
    logfile = logfile_;
}

bool QueryStream::connectToServer( Configuration cfg )
{
    char localeAttr[]="deu";
    try
    {
        // Allocating environment handle
        execCritical(EXAAllocHandle(SQL_HANDLE_ENV, NULL, &henv));
        // Setting up environment localization
        execCritical(EXASetEnvAttr(henv, EXA_ENV_ATTR_SET_LOCALE_ALL, localeAttr, SQL_NTS));
        // Allocating handle for connection
        execCritical(EXAAllocHandle(SQL_HANDLE_DBC, henv, &hdbc));
        // Turning autocommit off... is this necessary?
        execCritical(EXASetConnectAttr(hdbc,   SQL_ATTR_AUTOCOMMIT,   (void*)SQL_AUTOCOMMIT_OFF,
SQL_NTS) );
        // Setting up clientname
        execCritical(EXASetConnectAttr(hdbc, EXA_CLIENT_NAME, (void*)myID.c_str(), SQL_NTS));
        // Connect to server
        execCritical(EXAServerConnect(hdbc,  cfg.host,  SQL_NTS,  cfg.port,  SQL_NTS,  cfg.user,
SQL_NTS, cfg.password, SQL_NTS), henv);
    }
    catch(...)
    {
        // It would be nice to clean up handles
        return false;
    }
    return true;
}

bool QueryStream::prepare( Handle &handle, std::string statement )
{
          handle.max_size = 0;
          try
          {
                    // allocate statement handle
                    execCritical(EXAAllocHandle(SQL_HANDLE_STMT, hdbc, &handle.hstmt));
                    // enable column-wise binding
                    //execCritical(EXASetStmtAttr(                    SQL_ATTR_PARAM_BIND_TYPE,
SQL_PARAMETER_BIND_BY_COLUMN, 0 ))
                    // call prepare
                    execCritical(EXAPrepare(handle.hstmt,          (SQLCHAR*)statement.c_str(),
statement.size()) );
          }
          catch (...)
          {
                    if ( logfile ) LOG << "Error preparing statement: " << statement << endl;
                    cerr << "Error preparing statement: " << statement << endl;
                    return false;
          }
          return true;
}
```

```
void QueryStream::setSize( Handle &handle, long long int new_size )
{
        // set the number of rows in the parameter
        execCritical(  EXASetStmtAttr(  handle.hstmt,  SQL_ATTR_PARAMSET_SIZE,  (SQLPOINTER)
new_size, 0 ), handle.hstmt );
}

bool QueryStream::bindIntParameter( Handle &handle, uint *parameterArray, size_t size )
{
        // setting size intern
        handle.max_size = size;

        try
        {
                // binding parameter, setting sizes etc
                setSize( handle, size );
                //handle.statusArray = (SQLUSMALLINT*) malloc( size * sizeof( SQLUSMALLINT)
);
                handle.indArray = (SQLINTEGER*) malloc( size * sizeof( SQLINTEGER ) );
                for ( int i=0; i<size; i++ )
                {
                        handle.indArray[i] = 0;
                }
                // execCritical(  EXASetStmtAttr(  handle.hstmt,  SQL_ATTR_PARAM_STATUS_PTR,
handle.statusArray, 0 ) );
                execCritical(  EXASetStmtAttr(  handle.hstmt,  SQL_ATTR_PARAMS_PROCESSED_PTR,
&handle.parameterProcessed, 0 ) );
                execCritical(   EXABindParameter(   handle.hstmt,   1,   SQL_PARAM_INPUT,
SQL_C_ULONG, SQL_BIGINT, 0, 0, (SQLPOINTER) parameterArray, sizeof(uint),handle.indArray  ) );
        }
        catch (...)
        {
                if ( logfile ) LOG << "Error binding parameter" << endl;
                cerr << "Error binding parameter" << endl;
                return false;
        }
        return true;
}

bool QueryStream::execute( Handle &handle, size_t currentSize )
{
        // if necessary resize parameters (just smaller allowed)
        // call execute
        // check return values
    try
    {
                // rebinding if necessary
                if ( currentSize < handle.max_size )
                {
                        setSize( handle, currentSize );
                }
                //cout << "execute" << endl;
                execCritical( EXAExecute( handle.hstmt ), handle.hstmt );
                /*for ( int i=0; i<currentSize; i++ )
                {
                        cout << "i=" << i << endl;
                        execCritical( handle.statusArray[i], handle.hstmt );
                }*/
                //cout << "after execute" << endl;
    }
    catch(...)
    {
        if ( logfile ) LOG << "ERROR EXECUTING PREPARED STATEMENT\n\n";
        cerr << "\nERROR EXECUTING PREPARED STATEMENT\n" << endl;
                return false;
    }
        return true;
}

bool QueryStream::done( Handle &handle )
{
        // deallocate handle
        try
        {
                if ( handle.indArray != 0 )
                {
                        free( handle.indArray );
                        handle.indArray = 0;
                }
                /*if ( handle.statusArray != 0 )
                {
                        free( handle.statusArray );
                        handle.statusArray = 0;
                }*/
       execCritical(EXAFreeHandle(SQL_HANDLE_STMT, handle.hstmt));
        }
        catch (...)
        {
                cerr << "error in done" << endl;
                return false;
        }
        return true;
}

bool QueryStream::disconnectFromServer()
{
    try
    {
        // disconnecting from server
        execCritical(EXADisconnect(hdbc));
        //cout << "Successfully disconnected from server." << endl;
        // freeing connection handle
        execCritical(EXAFreeHandle(SQL_HANDLE_DBC, hdbc));
        //cout << "Freeing connection handle: ok" << endl;
        // freeing environment handle
                execCritical(EXAFreeHandle(SQL_HANDLE_ENV, henv));
        //cout << "Freeing environment handle: ok" << endl;
    }
    catch(...)
    {
        cerr << "Error while disconnecting from server" << endl;
        return false;
    }
    return true;
}



# ------------------------------------------------------------------------------------
# tpchquerystreammain.cpp
# ------------------------------------------------------------------------------------
#include "querystream.h"

#include <iostream>
using std::cout;
using std::endl;

#include <fstream>
using std::ofstream;
```

```
using std::ifstream;
using std::ostream;

#include <string>
using std::string;

#include <sstream>
using std::ostringstream;

#include <iomanip>

#define MAX_LINE_SIZE 10000

#include <sys/time.h>

/** \brief Calculates the time passed between t1 and t2. */
double timediff( timeval t1, timeval t2 )
{
    double passed = 1.0e-6 * ( t2.tv_usec - t1.tv_usec );
    passed += ( t2.tv_sec - t1.tv_sec );
    if ( passed < 0 )
        passed = -passed;
    return passed;
}

/** \brief Deletes from a given string all occurences of another string at beginning and end.
*/
string trim( string &s, const string drop = " " )
{
    string r = s.erase( s.find_last_not_of(drop)+1 );
    return r.erase( 0, r.find_first_not_of(drop) );
}

void timestamp( timeval mytime, ostream &out )
{
          struct tm tm_time;
    localtime_r( &mytime.tv_sec, &tm_time );
    int year = 1900 + tm_time.tm_year;
          // month in 0 (JAN)..11 (DEC). Add one to get the number of the month
    int month = tm_time.tm_mon+1;
    int day = tm_time.tm_mday;
    int hour = tm_time.tm_hour;
    int minute = tm_time.tm_min;
    int seconds = tm_time.tm_sec;
          int milisec = mytime.tv_usec / 1000;
    out  <<  std::setw(2)  <<  std::setfill('0')  <<  month  <<  "/"  <<  std::setw(2)  <<
std::setfill('0')  <<  day  <<  "/"  <<  std::setw(4)  <<  std::setfill('0')  <<  year  <<  "  "  <<
std::setw(2) << std::setfill('0') << hour << ":" << std::setw(2) << std::setfill('0') << minute
<<  ":"  <<  std::setw(2)  <<  std::setfill('0')  <<  seconds  <<  "."  <<  std::setw(3)  <<
std::setfill('0') << milisec;
}



int main( int argc, char** argv )
{
    timeval stream_starttime;
    timeval stream_endtime;
    timeval starttime;
    timeval endtime;

    gettimeofday(&stream_starttime, 0);

    bool firsttime = true;
    if ( argc != 7 && argc!=8 && argc != 9 && argc != 10 )
    {
        cout << "usage: " << argv[0] << " <inputfile> <logfile> <shortlog> <streamid> <host>
<port> [<schema>] [<user> <password>]" << endl;
        cout << endl;
        cout << "\tinputfile name of the file containing the queries to execute" << endl;
        cout << "\tlogfile   name of file to print output to" << endl;
                cout << "\tshortlog  name of file to print a very short version of the log
to (just times)" << endl;
        cout << "\tstreamid  an identifier used for this stream" << endl;
        cout << "\thost      name of host to connect to" << endl;
        cout << "\tport      port number on host to connect to" << endl;
                cout << "\tschema     name of a schema to open prior to any statements in
inputfile (default: open no schema)" << endl;
        cout << "\tuser      name of user to connect to EXASolution (default: sys)" << endl;
        cout << "\tpassword  password used to authenticate user (default: exasol)" << endl;
        return 1;
    }
    Configuration config;
    config.host = argv[5];
    config.port = argv[6];
    if ( argc == 9 )
    {
        config.user     = argv[7];
        config.password = argv[8];
    }
    else if ( argc == 10 )
      {
                config.user     = argv[8];
                config.password = argv[9];
      }
      else
    {
        config.user = "sys";
        config.password = "exasol";
    }
      std::string initialSchema = "";
      if ( argc == 8 || argc == 10 )
      {
                initialSchema = argv[7];
      }
    std::string myStreamID = argv[4];

    ofstream logfile( argv[2] );

      logfile.setf(std::ios::fixed);
      logfile.precision(2);

    logfile << "============================";
    for ( int i=0; i< myStreamID.size(); i++ )
        logfile << "=";
    logfile << '\n';
    logfile << "= Logfile for query stream " << myStreamID << " =\n";
    logfile << "============================";
    for ( int i=0; i< myStreamID.size(); i++ )
        logfile << "=";
    logfile << "\n\n";

      ofstream shortlog( argv[3] );
    shortlog << "Logfile for query stream " << myStreamID << " \n\n";
      shortlog << "Query; Start; End; Time\n";

    ifstream inputfile( argv[1] );
    QueryStream qs( myStreamID, &logfile );

    if ( qs.connectToServer( config ) )
```

```
{
    //cout << "MAIN: Connected to server" << endl;
}
else
{
    //cout << "MAIN: Error connecting to server. Trying to disconnect." << endl;
    qs.disconnectFromServer();
    return -1;
}

      // open the selected schema if any was given
      if ( initialSchema != "" )
      {
                std::string openSchemaStatement = "open schema " + initialSchema +";";
                qs.execute(openSchemaStatement);
      }

string queryText = "";
string currentLine;
while ( !inputfile.eof() )
{
    getline( inputfile, currentLine );
    currentLine = trim( currentLine );

    if ( currentLine.substr( 0, 2 ) == "--" )
    // comment
    {
        if ( currentLine.substr( 3, 11 ) == "TPC-H Query" )
        {
            if ( !firsttime )
            {
                gettimeofday( &endtime, 0 );
                logfile << "\n-----------\n";
                logfile << "- Summary -\n";
                logfile << "-----------\n\n";
                logfile << "started  at: " << ctime(&(starttime.tv_sec))
                        << " (epoch: " << starttime.tv_sec << " s, " << starttime.tv_usec
                        << " usec" << ")\n";
                logfile << "finished at: " << ctime(&(endtime.tv_sec))
                        << " (epoch: " << endtime.tv_sec << " s, " << endtime.tv_usec
                        << " usec" << ")\n";
                logfile << "time used  : " << timediff(starttime, endtime) << " s\n\n";

                shortlog << "; "; timestamp(starttime, shortlog);
                shortlog << "; "; timestamp(endtime, shortlog);
                shortlog << "; " << timediff(starttime, endtime) << "\n";

                starttime = endtime;
            }
            else
            {
                gettimeofday( &starttime, 0 );
                firsttime = false;
            }

            for ( int i=0; i<currentLine.size()-3 + 4; i++ )
                logfile << "*";
            logfile << '\n';
            logfile << "* " << currentLine.substr( 3 ) << " *" << endl;
            for ( int i=0; i<currentLine.size()-3 + 4; i++ )
                logfile << "*";
            logfile << "\n\n";

                                  shortlog << "Q" << currentLine.substr(15,2) << " ";
        }
            else if ( currentLine.substr( 3, 9 ) == "TPC-H End" )
                             // optional. If no TPC-H End is found, the end is at the end of
the input stream
            {
                break;
            }
            else
                logfile << currentLine << '\n';

            queryText = "";
            continue;
        }
        else if ( currentLine.size() == 0 )
        {
            logfile << '\n';
            continue;
        }
        else
        {
            queryText = queryText + currentLine + '\n';
            if ( currentLine[currentLine.size()-1] == ';' )
            {
                if ( !qs.execute( queryText, 1000 ) )
                                      // an error occured on execution
                                      {
                                                logfile                              <<
"!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n";
                                                logfile << "! Error(s) occured while executing
last statement. Exiting !\n";
                                                logfile                              <<
"!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n";

                                                shortlog                             <<
"!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n";
                                                shortlog   <<   "!   Error(s)   occured   while
executing last statement. Exiting !\n";
                                                shortlog                             <<
"!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n";

                                                qs.disconnectFromServer();
                                                return -1;

                                      }
                queryText = "";
            }
        }
    }
         // if any TPC-H Query was found, measure the time for the last one
         // (for all but the last query time measurement is done at the
         //  beginning of the next TPC-H query)
         if ( !firsttime )
         {
        gettimeofday( &endtime, 0 );
        logfile << "\n-----------\n";
        logfile << "- Summary -\n";
        logfile << "-----------\n\n";
        logfile << "started  at: " << ctime(&(starttime.tv_sec))
                << " (epoch: " << starttime.tv_sec << " s, " << starttime.tv_usec
                << " usec"<< ")\n";
        logfile << "finished at: " << ctime(&(endtime.tv_sec))
                << " (epoch: " << endtime.tv_sec << " s, " << endtime.tv_usec
                << " usec" << ")\n";
        logfile << "time used  : " << timediff(starttime, endtime) << " s\n\n";

        shortlog << "; "; timestamp(starttime, shortlog);
```

```
        shortlog << "; "; timestamp(endtime, shortlog);
        shortlog << "; " << timediff(starttime, endtime) << "\n";

        logfile << "*************\n";
        logfile << "* TPC-H End *\n";
        logfile << "*************\n\n";
          }

    //cout << "MAIN: everything done... exiting" << endl;
    if ( qs.disconnectFromServer() )
    {
        //cout << "MAIN: successfully disconnected." << endl;
    }
    else
    {
        //cout << "MAIN: Error while disconnecting." << endl;
    }
    gettimeofday( &stream_endtime, 0 );
    logfile << "--------------------------";
    for ( int i=0; i<myStreamID.size(); i++ )
        logfile << "-";
    logfile << "\n- Summary of querystream " << myStreamID << " -\n";
    logfile << "--------------------------";
    for ( int i=0; i<myStreamID.size(); i++ )
        logfile << "-";
    logfile << "\n\nstarted  at: " << ctime(&(stream_starttime.tv_sec))
            << " (epoch: " << stream_starttime.tv_sec << " s, "
            << stream_starttime.tv_usec << " usec"<< ")\n";
    logfile << "finished at: " << ctime(&(stream_endtime.tv_sec))
            << " (epoch: " << stream_endtime.tv_sec << " s, "
            << stream_endtime.tv_usec << " usec" << ")\n";
    logfile << "time used  : " << timediff(stream_starttime, stream_endtime)
            << " s\n\n";

    shortlog << "summary ";
    shortlog << "; "; timestamp(stream_starttime, shortlog);
    shortlog << "; "; timestamp(stream_endtime, shortlog);
    shortlog << "; " << timediff(stream_starttime, stream_endtime) << "\n";

    return 0;
}



# ------------------------------------------------------------------------------------------
# querystream.h
# ------------------------------------------------------------------------------------------
#ifndef QUERYSTREAM_H
#define QUERYSTREAM_H

#include <string>
#include <iostream>

#include "exaCInterface.h"

/** \brief The maximum length of a column name. */
static const size_t maxColumnNameSize = 40;

/** \brief A structure to describe connection information to a database.
  * Used in QueryStream::connectToServer().
  * \sa QueryStream::connectToServer() */
struct Configuration
{
    /** \brief The hostname of the server. */
    char *host;
    /** \brief The port of the server to connect to. */
    char *port;
    /** \brief The user name used for authentication to the database. */
    char *user;
    /** \brief The password of the user. */
    char *password;
};

/** \brief A structure used to fetch data from a database.
  * Each database column is represented by one FetchStruct.
  * Only one of the arrays of ODBC-C-datatypes is initialized and filled per column.
  * That one is identified by the c_datatype. */
struct FetchStruct
{
    /** \brief Array for columns of SQLCHAR* type
     *(one large block, easy access by sqlCharArrayAccess). */
    SQLCHAR        *sqlCharArray;
    /** \brief To quickly access the i-th string in the sqlCharArray array. */
          SQLCHAR*           *sqlCharArrayAccess;
    /** \brief Array for columns of SQLSMALLINT type. */
    SQLSMALLINT     *sqlSmallInt;
    /** \brief Array for columns of SQLUSMALLINT type. */
    SQLUSMALLINT    *sqlUSmallInt;
    /** \brief Array for columns of SQLINTEGER type. */
    SQLINTEGER      *sqlInteger;
    /** \brief Array for columns of SQLUINTEGER type. */
    SQLUINTEGER     *sqlUInteger;
    /** \brief Array for columns of SQLBIGINT type. */
    SQLBIGINT       *sqlBigInt;
    /** \brief Array for columns of SQLUBIGINT type. */
    SQLUBIGINT      *sqlUBigInt;
    /** \brief Array for columns of SQLREAL type. */
    SQLREAL         *sqlReal;
    /** \brief Array for columns of SQLDOUBLE type. */
    SQLDOUBLE       *sqlDouble;
    /** \brief Array for columns of SQLCHAR type. */
    SQLCHAR         *sqlChar;
    /** \brief Array for columns of SQLSCHAR type. */
    SQLSCHAR        *sqlSChar;
    /** \brief Array for columns of SQL_DATE_STRUCT type. */
    SQL_DATE_STRUCT *sqlDate;
    /** \brief Array for columns of SQL_NUMERIC_STRUCT type. */
    SQL_NUMERIC_STRUCT *sqlNumeric;

    /** \brief The length of a returned data and an indicator whether it is NULL. */
    SQLLEN *resultLen;

    /** \brief The name of the column. */
    SQLCHAR columnName[maxColumnNameSize];
    /** \brief The length of the column name. */
    SQLSMALLINT nameLength;
    /** \brief The SQL datatype ID of the column. */
    SQLSMALLINT datatype;
    /** \brief The ODBC C datatype ID of the column.
      * Might be set by QueryStream::find_ODBC_C_Type( SQLSMALLINT sqlType ) */
    SQLSMALLINT c_datatype;
    /** \brief The size of the column. */
    SQLUINTEGER columnSize;
    /** \brief The number of decimal digits of the column. */
    SQLSMALLINT decimalDigits;
    /** \brief Whether the column allows NULL values. */
    SQLSMALLINT nullable;
};

/** \brief The class represents a query stream to a server.
```

```
  * The query stream can be used to send any query as text to the server and
  * retrieve the result. */
class QueryStream
{
public:
    /** \brief Constructor.
      * \param streamId The identifier used to identify this query stream.
      * \param logfile The stream to be used for logging. */
    QueryStream( std::string streamId, std::ostream* logfile = &std::cout );
    /** \brief Tries to establish a connection to a server.
      * \param configuration The connection and login information for the server.
      * \return true if the connection could be established, false otherwise. */
    bool connectToServer( Configuration configuration );
    /** \brief Executes a given statement or query.
      * \param statement The text of the statement or query to execute.
      * \param fetchSize The number of rows to fetch with a single fetch command
      * (bigger fetch size might reduce overhead, but increases memory usage).
      * \return true if the statement was executed successfully, false if some error
      * occured while executing the statement or fetching the result set. */
    bool execute( std::string statement, long int fetchSize = standardFetchBlockSize );
    /** \brief Closes the connection to the server.
      * \return true if the connection was closed successfully, false if errors occured. */
    bool disconnectFromServer();
protected:
    /** \brief The maximum size of a string. */
    static const size_t maxStringSize  = 200;
    /** \brief The number of rows to fetch in a single fetch command. */
    SQLINTEGER fetchBlockSize;
          /** \brief The number to use for fetchBlockSize if it is not
      * given as argument to execute(). */
           static const size_t standardFetchBlockSize = 100;

    /** \brief Binds a column to an entry in the fetchMe array.
      * This must be done before fetching a resultset.
      * This function arranges memory allocation and binding function.
      * \param colNo The number of the resultset column to bind to.
      * \param localColNo The index in the fetchMe array the resultset data shall be stored.
      * \param datatype The SQL datatype of the column to bind to. */
    void bindColumn( int colNo, int localColNo, SQLSMALLINT datatype );

    /** \brief Allocates memory for the buffers needed in binding resultset columns.
      * The array corresponding to the given datatype of the given index
      * of the fetchMe array is initialized to the size of fetchBlockSize.
      * If the datatype is a string datatype, the string buffer will have the
      * size maxStringSize.
      * \param localColNo The index in the fetchMe array the data shall be allocated for.
      * \param datatype The ODBC-C-datatype memory shall be allocated for. */
    void allocateMemoryForBind( int localColNo, SQLSMALLINT datatype );

    /** \brief Maps an SQL-datatype-ID to a ODBC-C-datatype-ID to which it can be converted.
      * This is needed for binding columns and fetching results from a remote database.
      * \param sqlType The SQL-datatype-ID whose corresonding ODBC-C-datatype-ID is wanted.
      * \return The ODBC-C-datatype-ID for the given SQL-datatype-ID */
    SQLSMALLINT find_ODBC_C_Type( SQLSMALLINT sqlType );

    /** \brief Format resultset header for output. */
    void outputHeader();

    /** \brief Formats fetched rows for output.
      * The data in fetchMe is printed for each column and all fetched rows. */
    void outputFetchedRows();

    /** \brief The stream identifier. */
    std::string myID;

    /** \brief The environment handle used. */
    SQLHENV  henv;
    /** \brief The connection handle used. */
    SQLHDBC  hdbc;
    /** \brief The only statement handle used. */
    SQLHSTMT hstmt;

    /** \brief The structure used to fetch result sets of queries. */
    FetchStruct *fetchMe;

    /** \brief The number of rows in a result set. */
    SQLSMALLINT numResultCols;

    /** \brief The number of rows fetched by last fetch command. */
    SQLUINTEGER rowsFetched;

    /** \brief The stream used for logging results.*/
    std::ostream* logfile;
};

#endif // QUERYSTREAM_H



# ------------------------------------------------------------------------------------
# querystream.cpp
# ------------------------------------------------------------------------------------
#include "querystream.h"
#include <iostream>

#ifdef DEBUG
#define DEBUGOUT( X ) cout << "DEBUG:" << x
#else
#define DEBUGOUT( X )
#endif

#define LOG *logfile

using std::cout;
using std::endl;
using std::cerr;

SQLRETURN execCritical(SQLRETURN retcode, void* handle = 0)
{
    switch(retcode)
    {
    case SQL_INVALID_HANDLE:
                cerr << "invalid handle" << endl;
        throw "ERROR use of invalid handle!\n";
        break;
    case SQL_ERROR:
                cerr << "SQL error" << endl;
        throw "An error occured while executing command!\n";
        break;
          case SQL_SUCCESS_WITH_INFO:
                cerr << "Problem: Has Info, which should not be" << endl;
                throw "Problem: Has Info, which should not be!\n";
          break;
          case SQL_STILL_EXECUTING:
                cerr << "Still executing. This should not be should only";
                cerr << "return on successful execution!" << endl;
                throw "Unexpectedly still executing!\n";
                break;
          case SQL_NO_DATA:
          case SQL_SUCCESS:
```

```
        break;
          default:
                    // should not reach here...
                    cerr << "unexpected return code: " << retcode << endl;
                    throw "unexpected return code!\n";
                    break;
    }
    return retcode;
}

QueryStream::QueryStream( std::string streamId, std::ostream* logfile_ )
{
    myID  = streamId;
    henv  = NULL;
    hdbc  = NULL;
    hstmt = NULL;
    logfile = logfile_;
}

bool QueryStream::connectToServer( Configuration cfg )
{
    char localeAttr[]="deu";
    try
    {
        // Allocating environment handle
        execCritical(EXAAllocHandle(SQL_HANDLE_ENV, NULL, &henv));
        // Setting up environment localization
        execCritical(EXASetEnvAttr(henv, EXA_ENV_ATTR_SET_LOCALE_ALL, localeAttr, SQL_NTS));
        // Allocating handle for connection
        execCritical(EXAAllocHandle(SQL_HANDLE_DBC, henv, &hdbc));
        // Turning autocommit off... is this necessary?
        execCritical(EXASetConnectAttr(hdbc,   SQL_ATTR_AUTOCOMMIT,   (void*)SQL_AUTOCOMMIT_OFF,
SQL_NTS) );
        // Setting up clientname
        execCritical(EXASetConnectAttr(hdbc, EXA_CLIENT_NAME, (void*)myID.c_str(), SQL_NTS));
        // Connect to server
        execCritical(EXAServerConnect(hdbc,  cfg.host,  SQL_NTS,  cfg.port,  SQL_NTS,  cfg.user,
SQL_NTS, cfg.password, SQL_NTS), henv);
    }
    catch(...)
    {
        // It would be nice to clean up handles
        return false;
    }
    return true;
}

bool QueryStream::execute( std::string statement, long int fetchSize )
{
    try
    {
         // setting the fetchBlockSize (transmitted to EXASolution later on)
         this->fetchBlockSize = fetchSize;
        // put the statement in the logfile
        LOG << statement << '\n';
        // Allocating statement handle
        execCritical(EXAAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt));
        DEBUGOUT( "Allocating statement handle: ok" << endl );
        SQLCHAR* statementtext = (SQLCHAR*) statement.c_str();
        // execute the statement / query
        execCritical(EXAExecDirect(hstmt, statementtext, statement.size() ) );
        // get the affected rows (only insert, update, delete)
        SQLINTEGER affectedrows;
        execCritical(EXARowCount(hstmt,&affectedrows));
        // get the number of columns in the resultset (if any)
        execCritical(EXANumResultCols(hstmt,&numResultCols));
        // no result columns means no query
        if ( numResultCols <= 0 )
            LOG << "affected rows: " << affectedrows << "\n\n";

        // if a resultset was constructed, retrieve and print it
        if ( numResultCols > 0 )
        // a resultset was constructed
        {
            LOG << '\n';
            // the number of rows in the resultSet
            int resultSetSize = 0;
            // setting number of rows to fetch with each fetch
            SQLUINTEGER fetchsize = fetchBlockSize;
            execCritical(EXASetStmtAttr(hstmt, SQL_ATTR_ROW_ARRAY_SIZE,
                                        (SQLPOINTER) fetchsize , 0 ));
            //cout << "Fetch block size (SQL_ATTR_ROW_ARRAY_SIZE) set to " << fetchsize <<
endl;
            // setting position to store the fetch size for a query
            execCritical(EXASetStmtAttr(hstmt, SQL_ATTR_ROWS_FETCHED_PTR,
                                        (SQLPOINTER) &rowsFetched, 0));

            // allocate the fetch struct to be large enough for the columns
            fetchMe = new FetchStruct[numResultCols];
            // set all pointers to 0 to make it easy to free the correct memory later
            for ( int i=0; i<numResultCols; i++ )
            {
                fetchMe[i].sqlCharArray = 0;
                                        fetchMe[i].sqlCharArrayAccess = 0;
                fetchMe[i].sqlSmallInt  = 0;
                fetchMe[i].sqlUSmallInt = 0;
                fetchMe[i].sqlInteger   = 0;
                fetchMe[i].sqlUInteger  = 0;
                fetchMe[i].sqlBigInt    = 0;
                fetchMe[i].sqlUBigInt   = 0;
                fetchMe[i].sqlReal      = 0;
                fetchMe[i].sqlDouble    = 0;
                fetchMe[i].sqlChar      = 0;
                fetchMe[i].sqlSChar     = 0;
                fetchMe[i].sqlDate      = 0;
                fetchMe[i].sqlNumeric   = 0;
                fetchMe[i].resultLen    = 0;
            }
            execCritical(EXASetStmtAttr(hstmt, SQL_ATTR_ROWS_FETCHED_PTR,
                                        (SQLPOINTER) &rowsFetched, 0));
            // determine the datatype of the column and additional information.
            for ( SQLSMALLINT colNr=0; colNr<numResultCols; colNr++ )
            {
                // Get column description for the column with number colNr
                EXADescribeCol( hstmt, colNr+1, fetchMe[colNr].columnName,
                                maxColumnNameSize, &(fetchMe[colNr].nameLength),
                                &(fetchMe[colNr].datatype), &(fetchMe[colNr].columnSize),
                                &(fetchMe[colNr].decimalDigits), &(fetchMe[colNr].nullable) );
                fetchMe[colNr].c_datatype = find_ODBC_C_Type( fetchMe[colNr].datatype );

                // bind the column to an entry in the fetchMe array
                bindColumn( colNr+1, colNr, fetchMe[colNr].c_datatype );
            }
            // output the resultset header
            outputHeader();
            // fetch the columns and print them to output
            do
            {
                // fetch columns
                execCritical(EXAFetch( hstmt ));
```

```
            // output columns
            outputFetchedRows();
            // correct the result set size
            resultSetSize += rowsFetched;
        } while ( rowsFetched == fetchBlockSize );
        // add the total resultset size to the logfile
        LOG << "\nresultset consists of " << resultSetSize << " rows in " << numResultCols
<< " columns\n\n";
        // free memory for column variables again
        for ( int i=0; i<numResultCols; i++ )
        {
            if ( fetchMe[i].sqlCharArray != 0 )
            {
                delete[] fetchMe[i].sqlCharArray;
                delete[] fetchMe[i].sqlCharArrayAccess;
            }
            if ( fetchMe[i].sqlSmallInt  != 0 ) delete[] fetchMe[i].sqlSmallInt;
            if ( fetchMe[i].sqlUSmallInt != 0 ) delete[] fetchMe[i].sqlUSmallInt;
            if ( fetchMe[i].sqlInteger   != 0 ) delete[] fetchMe[i].sqlInteger;
            if ( fetchMe[i].sqlUInteger  != 0 ) delete[] fetchMe[i].sqlUInteger;
            if ( fetchMe[i].sqlBigInt    != 0 ) delete[] fetchMe[i].sqlBigInt;
            if ( fetchMe[i].sqlUBigInt   != 0 ) delete[] fetchMe[i].sqlUBigInt;
            if ( fetchMe[i].sqlReal      != 0 ) delete[] fetchMe[i].sqlReal;
            if ( fetchMe[i].sqlDouble    != 0 ) delete[] fetchMe[i].sqlDouble;
            if ( fetchMe[i].sqlChar      != 0 ) delete[] fetchMe[i].sqlChar;
            if ( fetchMe[i].sqlSChar     != 0 ) delete[] fetchMe[i].sqlSChar;
            if ( fetchMe[i].sqlDate      != 0 ) delete[] fetchMe[i].sqlDate;
            if ( fetchMe[i].sqlNumeric   != 0 ) delete[] fetchMe[i].sqlNumeric;
            if ( fetchMe[i].resultLen    != 0 ) delete[] fetchMe[i].resultLen;
        }
        delete[] fetchMe;
    }
    // free the handle for the statement again
    execCritical(EXAFreeHandle( SQL_HANDLE_STMT, hstmt ));
}
catch(...)
{
    LOG << "ERROR EXECUTING STATEMENT\n\n";
    cerr << "\nERROR EXECUTING STATEMENT:\n" << statement << '\n' << endl;
            return false;
}
      return true;
}

bool QueryStream::disconnectFromServer()
{
    try
    {
        // disconnecting from server
        execCritical(EXADisconnect(hdbc));
        //cout << "Successfully disconnected from server." << endl;
        // freeing connection handle
        execCritical(EXAFreeHandle(SQL_HANDLE_DBC, hdbc));
        //cout << "Freeing connection handle: ok" << endl;
        // freeing environment handle
        execCritical(EXAFreeHandle(SQL_HANDLE_ENV, henv));
        //cout << "Freeing environment handle: ok" << endl;
    }
    catch(...)
    {
        cerr << "Error while disconnecting from server" << endl;
        return false;
    }
    return true;
}
```

```
void QueryStream::outputHeader()
{
    int headerLength = 1;
    LOG << "|";
    for ( size_t col=0; col<numResultCols; col++ )
    {
        LOG << fetchMe[col].columnName << "|";
        headerLength += fetchMe[col].nameLength + 1;
    }
    LOG << '\n';
    for ( int i=0; i<headerLength; i++ )
        LOG << '=';
    LOG << '\n';
}

void QueryStream::outputFetchedRows()
{
    for ( size_t rows=0; rows<rowsFetched; rows++ )
    {
        LOG << "|";
        for ( size_t cols=0; cols<numResultCols; cols++ )
        {
            if ( fetchMe[cols].resultLen[rows] == SQL_NULL_DATA )
            {
                LOG << "|";
                continue;
            }
            switch( fetchMe[cols].c_datatype )
            {
                // string data
                case SQL_C_CHAR:
                case SQL_C_BINARY:
                //case SQL_C_XML:
                //case SQL_C_VARBOOKMARK:
                    LOG << fetchMe[cols].sqlCharArrayAccess[rows] << "|";
                break;

                // small int
                case SQL_C_SSHORT:
                    LOG << fetchMe[cols].sqlSmallInt[rows] << "|";
                break;

                // unsigned small int
                case SQL_C_USHORT:
                    LOG << fetchMe[cols].sqlUSmallInt[rows] << "|";
                break;

                // long int
                case SQL_C_SLONG:
                    LOG << fetchMe[cols].sqlInteger[rows] << "|";
                break;

                // unsigned long int
                case SQL_C_ULONG:
                    LOG << fetchMe[cols].sqlUInteger[rows] << "|";
                break;

                // float
                case SQL_C_FLOAT:
                    LOG << fetchMe[cols].sqlReal[rows] << "|";
                break;

                // double
                case SQL_C_DOUBLE:
```

```
    LOG << fetchMe[cols].sqlDouble[rows] << "|";
break;

// unsigned char
case SQL_C_BIT:
case SQL_C_UTINYINT:
    LOG << fetchMe[cols].sqlChar[rows] << "|";
break;

// signed char
case SQL_C_STINYINT:
    LOG << fetchMe[cols].sqlSChar[rows] << "|";
break;

// big int
case SQL_C_SBIGINT:
    LOG << fetchMe[cols].sqlBigInt[rows] << "|";
break;

// unsigned big int
case SQL_C_UBIGINT:
    LOG << fetchMe[cols].sqlUBigInt[rows] << "|";
break;

// date
case SQL_C_TYPE_DATE:
{
    // format date to YYYY-MM-DD for output
    if ( fetchMe[cols].sqlDate[rows].year < 1000 )
        LOG << "0";
    if ( fetchMe[cols].sqlDate[rows].year < 100 )
        LOG << "0";
    if ( fetchMe[cols].sqlDate[rows].year < 10 )
        LOG << "0";
    LOG << fetchMe[cols].sqlDate[rows].year << "-";

    if ( fetchMe[cols].sqlDate[rows].month < 10 )
        LOG << "0";
    LOG << fetchMe[cols].sqlDate[rows].month << "-";

    if ( fetchMe[cols].sqlDate[rows].day < 10 )
        LOG << "0";
    LOG << fetchMe[cols].sqlDate[rows].day << "|";
}
break;

// numeric, i.e. decimal
   // not used since decimal is mapped to double
case SQL_C_NUMERIC:
{
    // format decimal for output
    int scale = fetchMe[cols].sqlNumeric[rows].scale;
    char text[3*SQL_MAX_NUMERIC_LEN+2+scale];
    int pos_T = 3*SQL_MAX_NUMERIC_LEN+scale;
    int pos_N = SQL_MAX_NUMERIC_LEN-1;
    text[3*SQL_MAX_NUMERIC_LEN+1+scale] = '\0';
    //int add = 0;
    //if ( fetchMe[cols].sqlNumeric[rows].sign == 0 )
    //    add=1;
    while ( fetchMe[cols].sqlNumeric[rows].val[pos_N] == 0 && pos_N > 0)
        pos_N--;

    while ( pos_N >= 0 )
    {
        int current = fetchMe[cols].sqlNumeric[rows].val[pos_N];
        int newVal = 0;
        int currentPos = pos_N;
        while ( currentPos >= 0 )
        {
            //if ( currentPos == 0 )
            //{
            //    current += add;
            //    add = 0;
            //}
            if ( current < 10 )
            {

                fetchMe[cols].sqlNumeric[rows].val[currentPos] = newVal;
                currentPos--;

                if ( currentPos >= 0 )
                {
                    current       =       current       *       256       +
fetchMe[cols].sqlNumeric[rows].val[currentPos];
                    newVal = 0;
                }
            }
            else
            {
                newVal = current / 10;
                current = current % 10;
            }
        }
        if ( fetchMe[cols].sqlNumeric[rows].val[pos_N] == 0 )
        {
            pos_N--;
        }
        text[pos_T--] = '0' + current;
        if ( scale == 1 )
        {
            text[pos_T--] = '.';
        }
        scale--;
    }
    if ( scale==0)
        text[pos_T--] = '0';
    while ( scale > 0 )
    {
        text[pos_T--] = '0';
        if ( scale == 1 )
        {
            text[pos_T--] = '.';
            text[pos_T--] = '0';
        }
        scale--;
    }

    if ( fetchMe[cols].sqlNumeric[rows].sign == 0 )
        text[pos_T--] = '-';
    LOG << text+pos_T+1 << "|";
}
break;

// other datatype
default:
    cout << "output: datatype " << fetchMe[cols].c_datatype << " not supported
yet" << endl;
    throw fetchMe[cols].c_datatype;
break;
}
```

```
        }
        LOG << '\n';
    }
}

void QueryStream::bindColumn( int colNo, int localColNo, SQLSMALLINT c_datatype )
{
    // allocate memory for the column to bind
    allocateMemoryForBind( localColNo, c_datatype );
    // bind the column
    switch ( c_datatype )
    {
        // string data
        case SQL_C_CHAR:
        case SQL_C_BINARY:
        //case SQL_C_XML:
        //case SQL_C_VARBOOKMARK:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlCharArray,
                    maxStringSize*sizeof(SQLCHAR), fetchMe[localColNo].resultLen));
            DEBUGOUT( "bound C-String to column " << colNo << " (index " << localColNo << ")"
<< endl );
        break;

        // small int
        case SQL_C_SSHORT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlSmallInt,
                    sizeof(SQLSMALLINT), fetchMe[localColNo].resultLen));
            DEBUGOUT( "bound short int to column " << colNo << " (index " << localColNo << ")"
<< endl );
        break;

        // unsigned small int
        case SQL_C_USHORT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlUSmallInt,
                    sizeof(SQLUSMALLINT), fetchMe[localColNo].resultLen));
            DEBUGOUT( "bound unsigned short int to column " << colNo << " (index " <<
localColNo << ")" << endl );
        break;

        // long int
        case SQL_C_SLONG:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlInteger,
                    sizeof(SQLINTEGER), fetchMe[localColNo].resultLen));
            DEBUGOUT( "bound long int to column " << colNo << " (index " << localColNo << ")"
<< endl );
        break;

        // unsigned long int
        case SQL_C_ULONG:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlUInteger,
                    sizeof(SQLUINTEGER), fetchMe[localColNo].resultLen));
            DEBUGOUT( "bound unsigned long int to column " << colNo << " (index " << localColNo
<< ")" << endl );
        break;

        // float
        case SQL_C_FLOAT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlReal,
                    sizeof(SQLREAL), fetchMe[localColNo].resultLen));
            DEBUGOUT( "bound float to column " << colNo << " (index " << localColNo << ")" <<
endl );
        break;

        // double
        case SQL_C_DOUBLE:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlDouble,
                    sizeof(SQLDOUBLE), fetchMe[localColNo].resultLen));
            DEBUGOUT( "bound double to column " << colNo << " (index " << localColNo << ")" <<
endl );
        break;

        // unsigned char
        case SQL_C_BIT:
        case SQL_C_UTINYINT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlChar,
                    sizeof(SQLCHAR), fetchMe[localColNo].resultLen));
            DEBUGOUT( "bound char to column " << colNo << " (index " << localColNo << ")" <<
endl );
        break;

        // signed char
        case SQL_C_STINYINT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlSmallInt,
                    sizeof(SQLSCHAR), fetchMe[localColNo].resultLen));
            DEBUGOUT( "bound signed char to column " << colNo << " (index " << localColNo <<
")" << endl );
        break;

        // big int
        case SQL_C_SBIGINT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlBigInt,
                    sizeof(SQLBIGINT), fetchMe[localColNo].resultLen));
            DEBUGOUT( "bound signed big int to column " << colNo << " (index " << localColNo <<
")" << endl );
        break;

        // unsigned big int
        case SQL_C_UBIGINT:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlUBigInt,
                    sizeof(SQLUBIGINT), fetchMe[localColNo].resultLen));
            DEBUGOUT( "bound unsigned big int to column " << colNo << " (index " << localColNo
<< ")" << endl );
        break;

        // date
        case SQL_C_TYPE_DATE:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlDate,
                    sizeof(SQL_DATE_STRUCT), fetchMe[localColNo].resultLen));
            DEBUGOUT( "bound date to column " << colNo << " (index " << localColNo << ")" <<
endl );
        break;

        // numeric, i.e. decimal
        case SQL_C_NUMERIC:
            execCritical(EXABindCol(
                    hstmt, colNo, c_datatype, (SQLPOINTER) fetchMe[localColNo].sqlNumeric,
                    sizeof(SQL_NUMERIC_STRUCT), fetchMe[localColNo].resultLen));
```

```
            DEBUGOUT( "bound numeric to column " << colNo << " (index " << localColNo << ")" <<
endl );
        break;

        // other datatype
        default:
            DEBUGOUT( "column binding: datatype " << c_datatype << " not supported yet" << endl
);
            throw c_datatype;
        break;
    }
}

void QueryStream::allocateMemoryForBind( int localColNo, SQLSMALLINT datatype )
{
    // allocate memory for resultLen field used to indicate NULL values (among others)
    fetchMe[localColNo].resultLen = new SQLLEN[fetchBlockSize];

    // allocate memory depending on the column datatype
    switch ( datatype )
    {
        // string data
        case SQL_C_CHAR:
        case SQL_C_BINARY:
        //case SQL_C_XML:
        //case SQL_C_VARBOOKMARK:
            fetchMe[localColNo].sqlCharArray = new SQLCHAR[fetchBlockSize*maxStringSize];
                        fetchMe[localColNo].sqlCharArrayAccess          =          new
SQLCHAR*[fetchBlockSize];
            for ( size_t i=0; i<fetchBlockSize; i++ )
            {
                fetchMe[localColNo].sqlCharArrayAccess[i]                                    =
fetchMe[localColNo].sqlCharArray+i*maxStringSize;
            }
        break;

        // small int
        case SQL_C_SSHORT:
            fetchMe[localColNo].sqlSmallInt = new SQLSMALLINT[fetchBlockSize];
        break;

        // unsigned small int
        case SQL_C_USHORT:
            fetchMe[localColNo].sqlUSmallInt = new SQLUSMALLINT[fetchBlockSize];
        break;

        // long int
        case SQL_C_SLONG:
            fetchMe[localColNo].sqlInteger = new SQLINTEGER[fetchBlockSize];
        break;

        // unsigned long int
        case SQL_C_ULONG:
            fetchMe[localColNo].sqlUInteger = new SQLUINTEGER[fetchBlockSize];
        break;

        // float
        case SQL_C_FLOAT:
            fetchMe[localColNo].sqlReal = new SQLREAL[fetchBlockSize];
        break;

        // double
        case SQL_C_DOUBLE:
            fetchMe[localColNo].sqlDouble = new SQLDOUBLE[fetchBlockSize];
        break;

        // unsigned char
        case SQL_C_BIT:
        case SQL_C_UTINYINT:
            fetchMe[localColNo].sqlChar = new SQLCHAR[fetchBlockSize];
        break;

        // signed char
        case SQL_C_STINYINT:
            fetchMe[localColNo].sqlSChar = new SQLSCHAR[fetchBlockSize];
        break;

        // big int
        case SQL_C_SBIGINT:
            fetchMe[localColNo].sqlBigInt = new SQLBIGINT[fetchBlockSize];
        break;

        // unsigned big int
        case SQL_C_UBIGINT:
            fetchMe[localColNo].sqlUBigInt = new SQLUBIGINT[fetchBlockSize];
        break;

        // date
        case SQL_C_TYPE_DATE:
            fetchMe[localColNo].sqlDate = new SQL_DATE_STRUCT[fetchBlockSize];
        break;

        // numeric, i.e. decimal
        case SQL_C_NUMERIC:
            fetchMe[localColNo].sqlNumeric = new SQL_NUMERIC_STRUCT[fetchBlockSize];
        break;

        // other datatype
        default:
            cout << "memory allocation: datatype " << datatype << " not supported yet" << endl;
            throw datatype;
        break;
    }
}

SQLSMALLINT QueryStream::find_ODBC_C_Type( SQLSMALLINT sqlType )
{
    switch( sqlType )
    {
        case SQL_CHAR:
        case SQL_VARCHAR:
        case SQL_LONGVARCHAR:
        case SQL_WCHAR:
        case SQL_WVARCHAR:
        case SQL_WLONGVARCHAR:
            return SQL_C_CHAR;
        break;

        case SQL_DECIMAL:
        case SQL_NUMERIC:
            //return SQL_C_NUMERIC;
            return SQL_C_DOUBLE;
            //return SQL_C_CHAR;
        break;

        case SQL_SMALLINT:
            return SQL_C_SSHORT;
        break;

        case SQL_INTEGER:
```

```
            return SQL_C_SLONG;
        break;

        case SQL_REAL:
        case SQL_FLOAT:
        case SQL_DOUBLE:
            return SQL_C_DOUBLE;
        break;

        case SQL_BIT:
            return SQL_C_BIT;
        break;

        case SQL_TINYINT:
            return SQL_C_STINYINT;
        break;

        case SQL_BIGINT:
            return SQL_C_SBIGINT;
        break;

        case SQL_BINARY:
        case SQL_VARBINARY:
        case SQL_LONGVARBINARY:
            return SQL_C_BINARY;
        break;

        case SQL_TYPE_DATE:
            return SQL_C_TYPE_DATE;
        break;

        default:
            cout << "unsupported datatype, don't know corresponding C type: " << sqlType <<
endl;
            throw sqlType;
        break;
    }
}
```

```
#-------------------------------------------------------------------------------------------
# Diff of DBGen-Files
# -------------------------------------------------------------------------------------------

# dss.h

462c462
< #define SEPARATOR '|' /* field spearator for generated flat files */
---
> #define SEPARATOR ',' /* field spearator for generated flat files */
482a483
> #define PR_HUGE_LAST(f, str)  dbg_print(DT_HUGE, f, (void *)str, 0, 0)


# makefile.suite

76c70
< CC      =
---
 > CC      = /usr/bin/gcc
82,84c76,78
< DATABASE=
< MACHINE =
< WORKLOAD =
---
 > DATABASE= DWA
 > MACHINE = LINUX
 > WORKLOAD = TPCH
88c82
< CFLAGS        = -O -DDBNAME=\"dss\" -D$(MACHINE) -D$(DATABASE)
-D$(WORKLOAD)
---
> CFLAGS        = -O1 -DDBNAME=\"dss\" -D$(MACHINE) -D$(DATABASE)
-D$(WORKLOAD) -DTERABYTE -DEOL_HANDLIN -DEOL_HANDLING

print.c

114c114
<                       fprintf(target, "%s", (char *)data);
---
>                       fprintf(target, "\"%s\"", (char *)data);
153c153
<                       fprintf(target, "%c", *(char *)data);
---
>                       fprintf(target, "\"%c\"", *(char *)data);
484c484
<                 PR_HUGE(dfp, &new);
---
>                 PR_HUGE_LAST(dfp, &new);



# tpcd.h

51a52,60
 > #ifdef DWA
 > #define GEN_QUERY_PLAN "-- GEN_QUERY_PLAN"
 > #define START_TRAN     ""
 > #define END_TRAN       "COMMIT;"
 > #define SET_OUTPUT     ""
 > #define SET_ROWCOUNT   "LIMIT %d\n"
 > #define SET_DBASE      "OPEN SCHEMA %s;\n"
 > #endif
 >
```

Appendix G: Pricing Information

EXASOL AG
Neumeyerstrasse 48
D-90411 Nürnberg
T +49 911 23991 0
F +49 911 23991 5241

March 31, 2008

CPI Computer Partner Handels GmbH
Kapellenstr. 11
D-85622 Feldkirchen

Here is the requested price list:

Product	Part Number	Price	Quantity	Discount	Extended Price
Cluster-Software EXASolution 2.0 for 26 nodes with 12 GB License for 3 years (EXACluster OS 1.3 included)	EXA-26N-12G	395,767 $	1	10%	356,190 $
Premium-Support 24x7 (3.9%)	EXA-SUP-P	15,435 $	1		15,435 $
Total					**371,625 $**

This quote is valid for 60 days.

EXASOL Pricing Contact:
Olga Sapozhnykova
sales@exasol.com