TPC BENCHMARK™ A

Standard Specification

10 November 1989

Transaction Processing Performance Council (TPC)

Administered by
Waterside Associates
39510 Paseo Padre Pkwy, Suite 350
Fremont, CA  94538, USA
Phone: (415) 792-2901
FAX: (415) 792-4748
e-mail: shanley@cup.portal.com
TPC MEMBERSHIP
(September 1990)

Arix
AT&T
Bull S.A.
Computer Associates
Control Data Corp.
Data General
Digital Equipment Corp.
Fujitsu America
Hewlett Packard
Hitachi Ltd.
IBM
ICL
Informix
Ingres
Intel Corp.
ITOM International Corp.
Mitsubishi Electric Corp.
NCR
NEC Corp.
OKI Electric Industry
Olivetti
Oracle
Prime Computer
Pyramid Technology
Sequent Computer
Sequoia Systems
Siemens A.G.
Software A.G.
Stratus Computer
Sun Microsystems
Sybase
Tandem Computers
Teradata Corp.
Unify Corp.
Unisys
Wang Laboratories

TPC Benchmark™ is a trademark of the Transaction Processing Performance Council.
# TABLE OF CONTENTS

TPC MEMBERSHIP ............................................................................................................................. 2  
CLAUSE 0: Preamble............................................................................................................................. 4  
CLAUSE 1: Transaction and Terminal Profiles .................................................................................. 5  
CLAUSE 2: Transaction System Properties ....................................................................................... 7  
  2.1 The ACID Properties .................................................................................................................... 7  
  2.2 Atomicity Requirements .............................................................................................................. 7  
  2.3 Consistency Requirements .......................................................................................................... 7  
  2.4 Isolation Requirements .............................................................................................................. 8  
  2.5 Durability Requirements .......................................................................................................... 9  
CLAUSE 3: Logical Database Design ................................................................................................. 11  
CLAUSE 4: Scaling Rules .................................................................................................................. 13  
CLAUSE 5: Distribution, Partitioning, & Message Generation ......................................................... 14  
CLAUSE 6: Response Time ............................................................................................................ 15  
CLAUSE 7: Duration of Test ........................................................................................................... 17  
CLAUSE 8: SUT, Driver, & Communications Definition ............................................................... 18  
  8.1 Models of the Target System ....................................................................................................... 18  
  8.2 Test Configuration ...................................................................................................................... 19  
  8.3 System Under Test (SUT) Definition .......................................................................................... 19  
  8.4 Driver Definition ....................................................................................................................... 19  
  8.5 Communications Interface Definitions ...................................................................................... 20  
  8.6 Further Requirements on the SUT and Driver System ............................................................. 20  
  8.7 Reporting Metrics ...................................................................................................................... 22  
CLAUSE 9: Pricing ............................................................................................................................. 23  
CLAUSE 10: Full Disclosure ............................................................................................................. 26  
CLAUSE 11: Audit ............................................................................................................................ 28  
APPENDIX A: Sample Implementation .......................................................................................... 30  
(Not part of standard)
**CLAUSE 0: Preamble**

TPC Benchmark A exercises the system components necessary to perform tasks associated with that class of on-line transaction processing (OLTP) environments emphasizing update-intensive database services. Such environments are characterized by:

- Multiple on-line terminal sessions
- Significant disk input/output
- Moderate system and application execution time
- Transaction integrity

This TPC Benchmark uses terminology and metrics which are similar to other past or future benchmarks, originated by the TPC or others. Such similarity in terminology does not in any way imply that results are comparable to benchmarks other than TPC Benchmark A.

The metrics used in TPC Benchmark A are throughput as measured in transactions per second (tps), subject to a response time constraint; and the associated price-per-tps.

TPC Benchmark A can be run in a wide area or local area network configuration. The throughput metrics are "tpsA-Local" and "tpsA-Wide" respectively. The wide area and local area throughput and price-performance metrics are different and cannot be compared.

This benchmark uses a single, simple, update-intensive transaction to load the system under test (SUT). Thus the workload is intended to reflect an OLTP application, but does not reflect the entire range of OLTP requirements typically characterized by multiple transaction types of varying complexities. The single transaction type provides a simple, repeatable unit of work, and is designed to exercise the key components of an OLTP system.

The extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC Benchmark A approximates the customer application. Relative performance of systems derived from TPC Benchmark A do not necessarily hold for other workloads or environments. Extrapolations to unlike environments are not recommended.

A full disclosure report of the implementation details, as specified in Clause 10, must be made available along with the report results.

Benchmark results are highly dependent upon workload, specific application requirements, and system design and implementation. Relative system performance will vary as a result of these and other factors. Therefore TPC Benchmark A should not be used as a substitute for a specific customer application benchmarking when critical capacity planning and/or product evaluation decisions are contemplated.
CLAUSE 1: Transaction and Terminal Profiles

1.1 The Application Environment

1.1.1 This benchmark is stated in terms of a hypothetical bank. The bank has one or more branches. Each branch has multiple tellers. The bank has many customers, each with an account. The database represents the cash position of each entity (branch, teller, and account) and a history of recent transactions run by the bank. The transaction represents the work done when a customer makes a deposit or a withdrawal against his account. The transaction is performed by a teller at some branch. These functions are enumerated in 1.2.

1.1.2 The database may be implemented using any commercially available database management system (DBMS), database server, file system, etc. The terms “file/table”, “record/row” and “field/column” are used in this document only as examples of physical and logical data structures.

1.1.3 Implementors of this benchmark are permitted many possible system designs, insofar as they adhere to the standard model described and pictorially illustrated in Clause 8.

1.1.4 The word “terminal” as used in this standard refers to the teller interface device. This may be an actual terminal or the keyboard/display portion of an intelligent processor such as a workstation (see 9.2.2.2).

1.2 The Transaction Profile

Read 100 bytes including Aid, Tid, Bid, Delta from terminal (see 1.3)

BEGIN TRANSACTION
    Update Account where Account_ID = Aid:
        Read Account_Balance from Account
        Set Account_Balance = Account_Balance + Delta
        Write Account_Balance to Account
    Write to History:
        Aid, Tid, Bid, Delta, Time_stamp
    Update Teller where Teller_ID = Tid:
        Set Teller_Balance = Teller_Balance + Delta
        Write Teller_Balance to Teller
    Update Branch where Branch_ID = Bid:
        Set Branch_Balance = Branch_Balance + Delta
        Write Branch_Balance to Branch
COMMIT TRANSACTION

Write 200 bytes including Aid, Tid, Bid, Delta, Account_Balance to terminal (see 1.3)

Aid (Account_ID), Tid (Teller_ID), and Bid (Branch_ID) are keys to the relevant records/rows (see Clause 3.2).

1.3 Terminal I/O

1.3.1 For each transaction, the originating terminal shall send (see Clause 8) at least 100 user-level alphanumeric data bytes organized as at least four distinct fields, including Account_ID, Teller_ID, Branch_ID, and Delta. Branch_ID in the input message is the identifier of the branch where the teller is located.

1.3.2 Each terminal shall receive from the SUT at least 200 user-level alphanumeric data bytes, organized as at least five distinct fields as follows: Account_ID, Teller_id, Branch_ID, Delta, and Account_Balance resulting from successful commit of the transaction.

Comment: It is the intent of this Clause that the account balance in the database be returned to the application, i.e., that the application retrieve the account balance.

1.3.3 No compression shall be used on the user-level data in the message coming from or going to the terminal.
1.3.4 Any field(s) other than pure padding field(s) transmitted either way between the RTE (Remote Terminal Emulator, see Clause 8) and SUT (System Under Test, see Clause 8) in addition to the mandatory fields specified above must be disclosed, and the purpose of such field(s) explained.

1.3.5 The generation of input message fields is detailed in Clause 5.

1.4 **Specific Non-Requirements**

1.4.1 The order of the data manipulations within the transaction bounds is immaterial, and is left to the latitude of the test sponsor, as long as the transaction profile is functionally equivalent to the one outlined in Clause 1.2.

1.4.2 The transaction profile does not require that the SUT (see Clause 8) return the teller and branch balances to the application program.
CLAUSE 2: Transaction System Properties

2.1 The ACID Properties

2.1.1 The ACID (Atomicity, Consistency, Isolation, and Durability) properties of transaction processing systems must be supported by the system under test during the running of this benchmark. It is the intent of this section to informally define the ACID properties and to specify a series of tests that must be performed to demonstrate that these properties are met.

2.1.2 No finite series of tests can prove that the ACID properties are fully supported. Passing the specified tests is a necessary, but not sufficient, condition for meeting the ACID requirements.

2.1.3 All mechanisms needed to insure full ACID properties must be enabled during both the measurement and test periods. For example, if the system under test relies on undo logs, then logging must be enabled even though no transactions are aborted during the measurement period. When this benchmark is implemented on a distributed system, tests must be performed to verify that home and remote transactions (See Clause 5 for the definition of home and remote transactions.), including remote transactions that are processed on two nodes, satisfy the ACID properties.

2.2 Atomicity Requirements

2.2.1 Atomicity Property Definition

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.

2.2.2 Atomicity Tests

2.2.2.1 Perform the standard TPC Benchmark A transaction (see Clause 1.2) for a randomly selected account and verify that the appropriate records have been changed in the Account, Branch, Teller, and History files/tables.

2.2.2.2 Perform the standard TPC Benchmark A transaction for a randomly selected account, substituting an ABORT of the transaction for the COMMIT of the transaction. Verify that the appropriate records have not been changed in the Account, Branch, Teller, and History files/tables.

2.3 Consistency Requirements

2.3.1 Consistency Property Definition

Consistency is the property of the application that requires any execution of a transaction to take the database from one consistent state to another.

2.3.2 Consistency Conditions

A consistent state for the TPC Benchmark A database is defined to exist when:

a) the sum of the account balances is equal to the sum of the teller balances, which is equal to the sum of the branch balances;

b) for all branches, the sum of the teller balances within a branch is equal to the branch balance;

c) the history file has one logical record added for each committed transaction, none for any aborted transaction, and the sum of the deltas in the records added to the history file equals the sum of the deltas for all committed transactions.

If data is replicated, each copy must not violate these conditions.

2.3.3 Consistency Tests

Due to the large size of the Account file/table, no test of its consistency is specified. To verify the consistency of the Branch, Teller, and History files, perform the following (2.3.3.1 through 2.3.3.3 are meant to be performed in sequence):
2.3.3.1 Verify that the Branch and Teller files are initially consistent by performing the following steps:
   Step 1: Determine the balance of each branch as reflected in the branch file.
   Step 2: For each branch, calculate the branch balance by summing the balances of the tellers associated with the branch.
   Step 3: Verify that the balance of each branch as obtained from Steps 1 and 2 is the same.

2.3.3.2 Verify that the Branch and Teller files are still consistent after applying transactions to the database by performing the following steps:
   Step 1: Compute the initial sum of the branch balances for later use.
   Step 2: Count the number of records in the History file and sum the deltas in the History file. (The file may be empty).
   Step 3: Using the standard driving mechanism, submit a number of standard TPC Benchmark A transactions equal to at least ten times the number of tellers and note the number of transactions that are reported as committed. For example, a 100 tps (1000 teller) system must submit at least 10,000 transactions. If the number of committed transactions is not equal to the number of submitted transactions, explain why.
   Step 4: Re-verify the consistency of the Branch and Teller files by repeating 2.3.3.1.
   Step 5: Compute the final sum of the branch balances for later use.

2.3.3.3 Verify that the History file is consistent by performing the following steps:
   Step 1: Count the number of records in the History file and sum the deltas.
   Step 2: Verify that the count equals the original count from 2.3.3.2, Step 2, plus the number of transactions reported as committed in 2.3.3.2, Step 3. (The History file should contain one record for each committed transaction and should not contain a record for any aborted transaction).
   Step 3: Verify that the difference between the sum of the final and initial deltas in the History file is equal to the difference between the sum of the final and initial branch balances.

2.4 Isolation Requirements

2.4.1 Isolation Property Definition

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.

This property is commonly called serializability. Sufficient conditions must be enabled at either the system or application level to ensure serializability of transactions under any mix of arbitrary transactions, not just TPC Benchmark A transactions. The system or application must have full serializability enabled, i.e., repeated reads of the same records within any committed transaction must have returned identical data when run concurrently with any mix of arbitrary transactions.

2.4.2 Isolation Tests

For conventional locking schemes, isolation should be tested as described below, where transactions 1 and 2 are versions of the standard TPC Benchmark A transaction. Systems that implement other isolation schemes may require different validation techniques. It is the responsibility of the test sponsor to disclose those techniques and the tests for them.

2.4.2.1 Isolation Test for Completed Transactions (conventional locking schemes):
   - Start transaction 1.
   - Stop transaction 1 immediately prior to COMMIT.
   - Start transaction 2.
   - Transaction 2 attempts to update the same account record as transaction 1.
   - Verify that transaction 2 waits.
   - Allow transaction 1 to complete. Transaction 2 should now complete.
   - Verify that the account balance reflects the results of both updates.
2.4.2.2 Isolation Test for Aborted Transactions (conventional locking schemes):

Start transaction 1.
Stop transaction 1 immediately prior to COMMIT.
Start transaction 2.
Transaction 2 attempts to update the same account record as transaction 1.
Verify that transaction 2 waits.
Abort transaction 1. Transaction 2 should now complete.
Verify that the account balance reflects the results of transaction 2's update only.

2.4.2.3 Repeat tests 2.4.2.1 and 2.4.2.2 for the branch and teller files.

2.5 Durability Requirements

The tested system must guarantee the ability to preserve the effects of committed transactions and insure database consistency after recovery from any one of the failures listed below in Clause 2.5.3.

**Comment:** No system provides complete durability, i.e., durability under all possible types of failures. The specific set of single failures addressed in 2.5.3 is deemed sufficiently significant to justify demonstration of durability across such failures.

2.5.1 Durable Medium Definition

A durable medium is a data storage medium that is either:

a) an inherently non-volatile medium, e.g., magnetic disk, magnetic tape, optical disk, etc., or
b) a volatile medium with its own self-contained power supply that will retain and permit the transfer of data, before any data is lost, to an inherently non-volatile medium after the failure of external power.

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

**Comment:** A durable medium can fail; this is usually protected against by replication on a second durable medium (e.g., mirroring) or logging to another durable medium. Memory can be considered a durable medium if it can preserve data long enough to satisfy the requirement stated in b) above. For example, if it is accompanied by an uninterruptible power supply, and the contents of memory can be transferred to an inherently non-volatile medium during the failure. Note that no distinction is made between main memory and memory performing similar permanent or temporary data storage in other parts of the system, e.g., disk controller caches.

2.5.2 Committed Property Definition

A transaction is considered committed when the transaction manager component of the system has written the commit record(s) associated with the transaction to a durable medium.

**Comment 1:** Transactions can be committed without the user subsequently receiving notification of that fact, since message integrity is not required for TPC Benchmark A.

**Comment 2:** Although the order of operations in the transaction profile (Clause 1.2) is immaterial, the actual transmission of the output message cannot begin until the commit operation has successfully completed.

2.5.3 List of single failures

2.5.3.1 Permanent irrecoverable failure of any single durable medium containing database, ABTH files/tables, or recovery log data.

**Comment:** If main memory is used as a durable medium, then it must be considered as a potential single point of failure. Sample mechanisms to survive single durable medium failures are: i) database archiving in conjunction with a redo (after image) log, and ii) mirrored durable media. If memory is the durable medium and mirroring is the mechanism used to ensure durability, then the mirrored memories must be independently powered.
2.5.3.2 Instantaneous interruption (system crash/system hang) in processing which requires system reboot to recover.

**Comment:** This implies abnormal system shutdown which requires loading of a fresh copy of the operating system from the boot device. It does not necessarily imply loss of volatile memory. When the recovery mechanism relies on the pre-failure contents of volatile memory, the means used to avoid the loss of volatile memory, e.g., uninterruptible power supply, must be included in the system cost calculation. A sample mechanism to survive an instantaneous interruption in processing is an undo/redo log.

2.5.3.3 Failure of all or part of memory (loss of contents).

**Comment:** This implies that all or part of memory has failed. This may be caused by a loss of external power or the permanent failure of a memory board.

2.5.4 The recovery mechanism cannot use the contents of the History file to support the durability property.

2.5.5 Rollforward recovery from an archive database copy, e.g., a copy taken prior to the run, using redo log data is not acceptable as the recovery mechanism in the case of failures listed in 2.5.3.2 and 2.5.3.3. Note that "checkpoints", "control points", "consistency points", etc. of the database taken during a run are not considered to be archives.

2.5.6 Durability Tests

The intent of these tests is to demonstrate that all transactions whose output messages have been received at the terminal or RTE have in fact been committed in spite of any single failure from the list in Clause 2.5.3.

It is not required to perform these tests under a full terminal load or with a fully scaled database. However, the test sponsor must state that to the best of their knowledge a fully loaded and fully scaled test would also pass the durability tests. It is required to use the same SUT configuration and database partitioning as was used in the measurement part of the test. Furthermore, at the time of the induced failures, it is required to have multiple home and remote transactions (See Clause 5) in progress and distributed systems must have distributed transactions in progress as well.

For each of the failure types defined in Clause 2.5.3, perform the following steps:

1. **Step 1:** Perform Step 1 of the History file Consistency Test in Clause 2.3.3.3.
2. **Step 2:** Start submitting TPC Benchmark A transactions. On the driver system, record committed transactions in a "success" file.
3. **Step 3:** Cause a failure selected from the list in Clause 2.5.3.
4. **Step 4:** Restart the system under test using normal recovery procedures.
5. **Step 5:** Compare the contents of the "success" file and the History file to verify that every record in the "success" file has a corresponding record in the History file. Also verify that the number of records in the History file is greater or equal to the original count, as obtained in Step 1, plus the number of records in the "success" file. If there is an inequality, the History file must contain additional records and the difference must be less than or equal to the number of terminals simulated. **Comment:** This difference should be due only to transactions which were committed on the system under test, but for which the 200 byte output message was not transmitted back to the driver before the failure.
6. **Step 6:** Perform the consistency test on the Branch and Teller files as specified in Clause 2.3.3.2.
**CLAUSE 3: Logical Database Design**

3.1  **Entities, Relationships, and Characteristics**

3.1.1  The components of the database are defined to consist of four separate and individual files/tables: Account, Branch, Teller, and History. The relationships among these files/tables are defined in the following entity/relationship diagram and are subject to the business rules specified in 3.1.2. This diagram is a logical description and has no implication for physical implementation.

**Comment:**
- The clustering of records within the database (as in hierarchical or CODASYL databases) is not excluded.
- A view which represents the records/rows to avoid read/writes is excluded.

![Entity Relationship Diagram]

3.1.2  The entities in 3.1.1 are subject to the following business rules:
- All branches must have the same number of tellers.
- All branches must have the same number of accounts.

Other business rules specified elsewhere in this document also apply, e.g., consistency conditions in Clause 2.3.2.

3.2  **Record Layouts and Sizing**

3.2.1  In order for the transaction to represent a similar amount of work to all the systems, it is important that the records handled by the database servers, file systems, etc. be of the same size. Therefore, the records/rows must be stored in an uncompressed format. Where it is impossible to turn compression off, it is incumbent upon the test sponsor to store the records/rows using the minimum lengths specified in 3.2.2 through 3.2.5. Any space with unspecified values in the record/row descriptions in 3.2.2 through 3.2.5 may be used for additional user data; the storage for the access path (e.g., B-tree index structure) or any other data used by the database server may not be counted against the minimum record length specifications.

3.2.2  Account records/rows must be at least 100 bytes in length and contain the following data in any order or representation:

- **Account_ID**  Must uniquely identify the record/row across the range of accounts. The Account_ID must be unique across the entire database.
- **Branch_ID**  Branch where account is held.
- **Account_Balance**  Must be capable of representing at least 10 significant decimal digits plus sign.

3.2.3  Branch records/rows must be at least 100 bytes in length and contain the following data in any order or representation:

- **Branch_ID**  Must uniquely identify the record/row across the range of branches.
- **Branch_Balance**  Must be capable of representing at least 10 significant decimal digits plus sign.
3.2.4 Teller records/rows must at least 100 bytes in length and contain the following data in any order or representation:

- **Teller_ID** Must uniquely identify the record/row across the range of tellers.
- **Branch_ID** Branch where the teller is located.
- **Teller_Balance** Must be capable of representing at least 10 significant decimal digits plus sign.

3.2.5 History records/rows must be at least 50 bytes in length and contain the following data in any order or representation:

- **Account_ID** Account updated by transaction.
- **Teller_ID** Teller involved in transaction.
- **Branch_ID** Branch associated with Teller.
- **Amount** Amount (delta) specified by transaction. Must be capable of representing at least 10 significant decimal digits plus sign.
- **Time_Stamp** A date and time taken between BEGIN TRANSACTION and COMMIT TRANSACTION. It must be capable of representing Date as YY:MM:DD and Time with a resolution of at least HH:MM:SS.

3.3 The size of the identifier in each record/row must be sufficient for the size of the configured system (see Clause 4.2). Thus for a 100 tps test, the accounts file/table must include 10 million records/rows, and hence the account identifier, i.e. the Account_ID, must be able to represent at least 10 million unique numbers.

3.4 The record identifiers of the Account/Branch/Teller (ABT) files/tables must not directly represent the physical disk addresses of the records or any offsets thereof. The application may not reference records using relative record numbers since they are simply offsets from the beginning of a file. This does not preclude hashing schemes or other file organizations which have provisions for adding, deleting, and modifying records in the ordinary course of processing. This Clause places no restrictions on the History file.

**Comment:** It is the intent of this Clause that the application executing the transaction not use physical identifiers, but logical identifiers for all accesses; i.e., it is not legitimate for the application to build a "translation table" of logical-to-physical addresses and use it for enhancing performance.

3.5 While inserts and deletes are not performed on the ABT files/tables, the SUT must not be configured to take special advantage of this fact.
**CLAUSE 4: Scaling Rules**

4.1 The intent of the scaling rules is to maintain a fixed relationship between the transaction load presented to the system under test and the size of the files/tables accessed by the transactions.

4.2 For each nominal transaction-per-second (tps) configured, the test must use a minimum of (see Clause 4.4):

<table>
<thead>
<tr>
<th>Category</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account records/rows</td>
<td>100,000</td>
</tr>
<tr>
<td>Teller records/rows</td>
<td>10</td>
</tr>
<tr>
<td>Branch records/rows</td>
<td>1</td>
</tr>
<tr>
<td>History record/rows</td>
<td>(See 4.3)</td>
</tr>
<tr>
<td>Terminals</td>
<td>1</td>
</tr>
</tbody>
</table>

4.2.1 All terminals should be active throughout the steady state period. The intent is that each terminal should contribute no more than $1/10$ tps per terminal, i.e. the minimum mean inter-arrival time must be 10 seconds. The distribution of transactions with respect to time is specified in Clause 8.6.3.

4.2.2 Should any value in 4.2 be exceeded, the others should be increased proportionately to maintain the same ratios among them as in 4.2. For example, if 200 terminals are used to generate 10 tps then there must be 20 branch records, 200 teller records, and 2,000,000 account records in the database and the price of the system must include 200 terminals (see Clause 9.1.2).

4.3 The history file/table should be large enough to hold all history data generated during the steady state portion of the test. However, for the purpose of computing price per tps, storage must be maintained for the number of history records specified in Clause 9.2.3.1. This includes the overhead space required to manage and access the data as well as data space. The system under test must be physically configurable to support the amount of storage specified in Clause 9.2.3.1.

4.4 Reported tps may not exceed the configured (nominal) rate represented by the file/table sizes in 4.2. While the reported tps may fall short of the maximum allowed by the configured system, the price per tps computation must report the price for the system as actually configured.
CLAUSE 5: Distribution, Partitioning, & Message Generation

5.1 Types of Transactions and Nodes

5.1.1 A transaction is home if the account is held at the same branch as the teller that is involved in the transaction (See Clause 3.1.1).

5.1.2 A transaction is remote if the branch where the account is held is not the same as the branch associated with the teller involved in the transaction.

5.1.3 A remote transaction may be processed entirely on a single-node or be distributed between two separate nodes. If the account branch and the teller branch exist on different nodes, the node containing the teller branch is referred to as the native node, and the node containing the account branch (the remote branch) is referred to as the foreign node.

5.2 Partitioning Rules

5.2.1 Horizontal partitioning of files/tables is allowed. For example, groups of history records/rows may be assigned to different files, disks or areas. If this partitioning is not transparent to the logic of the transaction program, it must be disclosed.

5.2.2 Vertical partitioning of files/tables is not allowed. For example, groups of fields/columns of one record/row may not be assigned to files, disks, or areas different from those storing the other fields/columns of that record/row. The record must be processed as a series of contiguous fields. Note: This restriction is included to normalize vendor benchmarks, since it is the intent of the standard that each TPC Benchmark A data operation accesses approximately 100 bytes, not some smaller, proper subset.

5.3 Input Message Generation

5.3.1 The input message fields (Account_ID, Branch_ID, Teller_ID, and Delta) must conform to the database fields definition of Clause 3.

5.3.2 The Branch_ID and Teller_ID are constant over the whole measurement period for any given terminal.

5.3.3 The Delta amount field is a random value within [-999999, +999999] selected independently for each transaction.

5.3.4 The Account_ID is generated as follows:
   • A random number X is generated within [0,1]
   • If X<0.85, a random Account_ID is selected over all <Branch_ID> accounts.
   • If X>=0.85, a random Account_ID is selected over all non-<Branch_ID> accounts.

Comment 1: This algorithm guarantees that an average of 15% of remote transactions is presented to the SUT. Due to statistical variations during a finite measurement period, the actual measured proportion of remote transactions may vary around 15%. As a rule of thumb, actual measured values should be within 14%-16% for any meaningful number of transactions.

Comment 2: In a distributed system, the 85-15 rule should be implemented so that the ratio of remote-branch transactions occurring on a foreign node is proportional to the actual distribution of accounts across the nodes. For example, if 3000 branches are divided evenly between two nodes, approximately 7.5% (1500/2999 * 15%) of the transactions cause cross-node activities. With the same 3000 branches divided among three nodes, approximately 10% (2000/2999 * 15%) cause cross-node activities, etc. Note that 2999 is used since the home branch by definition does not qualify.

5.4 "Random" Definition

Within Clause 5, the term random means independently selected and uniformly distributed.
CLAUSE 6: Response Time

6.1 Measurement Interval and Timing

6.1.1 In this Clause, the term "measurement interval" is the steady state period (see Clause 7.1) during the execution of the benchmark for which the test sponsor is reporting a tps number and response time data. The term "completed transaction" is a transaction which has been successfully committed at the SUT and whose output message has been recorded at the Remote Terminal Emulator (RTE; see Clause 8.4).

6.1.2 Each transaction submitted to the SUT must be individually timed.

6.2 Response Time Definition

Response times must be measured at the RTE. The response time (RT) of a transaction is defined by:

\[ RT = T2 - T1 \]

where \( T1 \) and \( T2 \) are measured at the RTE and defined as:

- \( T1 \) - time stamp taken before the first byte of the input message is sent from the RTE to the SUT.
- \( T2 \) - time stamp taken after the last byte of the output message from the SUT arrives at the RTE.

6.3 Response Time Constraint

90% of all transactions started and completed during the measurement interval must have a Response Time of less than 2 seconds.

Comment: This response time criteria has been chosen to provide a single criteria for all configurations, and in particular systems with wide-area network (WAN) communications, and very-low throughput systems.

6.4 Computation of tps Rating

6.4.1 The reported tps is the total number of committed transactions which both started and completed at the RTE during the measurement interval, divided by the elapsed time of the interval.

6.4.2 Reported tps must be expressed to a minimum precision of three significant digits, rounded down.

6.5 Interpolation, Extrapolation Prohibited

The reported tps rate must be measured rather than interpolated or extrapolated. For example, suppose 9.10 tps is measured on a 100 terminal test during which 90% of the transactions completed in less than 1.7 seconds and 9.7 tps is measured on a 110 terminal test during which 90% of the transactions completed in less than 2.3 seconds. Then the reported tps is 9.1 rather than some interpolated value between 9.1 and 9.7.

6.6 Required Reporting

6.6.1 The frequency distribution of response times of transactions started and completed during the measurement interval must be reported. The range of the X axis must be from 0 to 20 seconds response time. At least 20 different intervals, of equal one-second-or-less length, must be reported. A sample graph is shown below. The maximum and average response times must also be reported.
6.6.2 A complete curve of response times versus tps must be reported. The points on the curve must be of the form \((x, y)\), where:

\[
x = \text{measured tps}
\]

\[
y = \text{corresponding 90th percentile of response times}
\]

A curve must be plotted at approximately 50%, 80%, and 100% of reported throughput points (additional points are optional). The 50% and 80% points are to be measured on the same configuration as the 100% run, varying the think times. Interpolation of the curve between these data points below the 100% level is permitted. An example of such a curve is shown below.
CLAUSE 7: Duration of Test

7.1 Steady State

The test must be conducted in a "steady state" condition that represents the true "sustainable performance" of the system under test (SUT).

Although the measurement period as described below may be as short as 15 minutes, the system under test must be configured so that it is possible to run the test at the reported tps for a continuous period of at least eight hours. For example, the media used to store at least eight hours of log data must be configured.

7.2 Duration and Requirements

The measurement period must:

- Begin after the system reaches sustained "steady state";
- Be long enough to generate reproducible tps results;
- Extend uninterrupted for at least 15 minutes and no longer than 1 hour;
- For systems which defer database writes to durable media, recovery time from instantaneous interruptions (as defined in Clause 2.5.3.2) must not be appreciably longer at the end of the measurement period than at the beginning of the measurement period.

Comment 1: "Steady state" is easy to define, e.g., "sustained throughput," but difficult to prove. The test sponsor (and/or the auditor) is required to report the method used to verify steady state sustainable performance and the reproducibility of test results. The auditor is encouraged to use available monitoring tools to help determine steady state.

Comment 2: The intent of this Clause is to require that writes to disk or other durable media that would normally occur during a sustained test of at least eight hours duration (such as checkpointing, writing redo/undo log records to disk, etc.), are included in the measurement interval and are not deferred until after the measurement is complete.

Note to Comment 2: Some systems defer writes of changed pages/blocks to the durable-medium-resident database. Such systems can maintain buffers/caches in a volatile medium (e.g., memory) for use by the DBMS, operating system, and disk control system, which are not synchronized with the durable-medium-resident database. Re-synchronizing these caches with the durable-medium-resident database is typically accomplished via "control points," "checkpoints," or "consistency points."
CLAUSE 8: SUT, Driver, & Communications Definition

8.1 Models of the Target System

Models of the system which is the target (object) of this benchmark are shown pictorially below. By way of illustration, the diagrams also depict the RTE/ SUT boundary (see Clause 8.3 and 8.4) where the response time is measured.

If any network other than the Sever-Server network is a wide area network as defined in Clause 8.5.1, and at least one message of each transaction passes through the WAN, then the system is called a WAN-approach; otherwise, it is called a LAN-approach.

8.2 **Test Configuration**

The test configuration consists of the following elements:

- System Under Test (SUT)
- Driver System
- Driver/SUT Communications Interface

The tested configuration need not include the WAN long-haul communications lines.

8.3 **System Under Test (SUT) Definition**

The SUT consists of:

- One or more processing units (e.g., hosts, front-ends, workstations, etc.) which will run the transaction described in Clause 1, and whose aggregate performance will be described by the metric tpsA-Wide or tpsA-Local.
- Any front-end systems are considered to be part of the SUT. Examples of front-end systems are terminal demultiplexers, front-end data communications processors, cluster controllers, database clients (as in the 'client/server' model), and workstations.
- The hardware and software components of all networks required to connect and support the SUT components.
- Data storage media sufficient to satisfy both the scaling rules in Clause 4 and the ACID properties of Clause 2. The data storage media must hold all the data described in Clause 3 and be intimately attached to the processing units(s).
- The host system(s) including hardware and software supporting the database employed in the benchmark.

8.4 **Driver Definition**

8.4.1 An external Driver System, which provides Remote Terminal Emulator (RTE) functionality, will be used to emulate the target terminal population during the benchmark run. The terminal population is scaled in accordance with Clause 4.

8.4.2 The RTE:

- Generates and sends 100 byte transactional messages to the SUT;
- Receives 200 byte responses;
- Records message response times;
- Performs conversion and/ or multiplexing into the communications protocol used by the communications interface between the driver and the SUT;
- Statistical accounting is also considered a RTE function.

The possibility of utilizing an actual real-terminal configuration as an RTE is not excluded.

8.4.3 Normally, the Driver System is expected to perform RTE functions only. Work done on the Driver System in addition to the RTE as specified in 8.4.2 must be thoroughly justified as specified in Clause 8.6.4.

8.4.4 The intent is that the Driver System must reflect the proposed terminal configuration and cannot add functionality or performance above the priced network components in the SUT. It must be demonstrated that performance results are not enhanced by using a Driver System. (See Clause 10.1.7.2).

8.4.5 Any software or hardware which resides on the Driver which is not the RTE is to be considered as part of the SUT. For example, in a client-server model, the client software may be run or be simulated on the Driver system (see Clause 8.6.4).
8.5 Communications Interface Definitions

8.5.1 Wide Area Network (WAN) and Local Area Network (LAN) Definitions

8.5.1.1 A wide area network is defined as a communications interface capable of supporting remote sessions over a distance of at least 1500 kilometers, with a protocol supported by commercially available products.

8.5.1.2 The upper limit on WAN communications bandwidth will be 64 kbps (Kbits/second) per communications line utilized, and the number of terminals simulated over a 64 kbps line is restrained only by the bandwidth of that line.

Comment 1: The communications line will operate at 64 kbps at both ends (Terminal and SUT), but may utilize higher bandwidth mechanisms in between. A maximum line speed of 64 kbps has been selected because of global availability, thus ensuring that country metrics can be published.

Comment 2: In order for a network to be considered a WAN:

- At least one message for each transaction must pass through a WAN.
- All components of the WAN (e.g., modems, multiplexers, etc.) must be capable of operating over a distance of at least 1500 kilometers. This implies that timeouts, turnaround delays, etc., must be accounted for.

8.5.1.3 If a network is not a WAN, it is a Local Area Network (LAN).

8.5.1.4 All protocols used must be commercially available.

Comment: It is the intention of this definition to exclude non-standard I/O channel connections. The following situations are examples of acceptable channel connections:

- Configurations or architectures where terminals or terminal controllers are normally and routinely connected to an I/O channel of a processor.
- Where the processor(s) in the SUT is/are connected to the local communications network via a front-end processor which is channel connected. The front-end processor is priced as part of the SUT.

8.5.2 Driver/SUT Communications Interface

8.5.2.1 The communications interface between the Driver System and the SUT must be the mechanism by which the system would be connected with the end-user devices (terminals and/or workstations) in the proposed configuration.

8.6 Further Requirements on the SUT and Driver System

8.6.1 No Database on Driver System

Copies of any part of the tested data base or file system or its data structures, indices, etc. may not be present on the Driver System during the test. Synchronization between RTE and SUT (e.g., through known initial values for ABT balances) is equally disallowed.

8.6.2 Individual Contexts for Emulated Terminals

The SUT must contain context for each terminal emulated, and must maintain that context for the duration of that test. That context must be identical to the one which would support a real terminal. A terminal which sends a transaction, cannot send another until the completion of that transaction.

Comment: The 'context' referred to in 8.6.2 should consist of information such as terminal identification, network identification, and other information necessary for a real terminal to be known to (i.e. configured on) the SUT. The intention is to allow pseudo-conversational transactions. The intent of 8.6.2 is simply to prevent a test sponsor from multiplexing messages from a very large number of emulated terminals into a few input lines and claiming or implying that the tested system supports that number of users regardless of whether the system actually supports that number of real terminals.
8.6.3 Pacing of Transactions by Emulated Terminals

Each emulated terminal, after sending a request to update the database to the SUT, must wait for a given "Think Time" after receiving that reply, before sending the next request. By definition, the Response Time added to the Think Time gives the Cycle Time, which has to average at least 10 seconds (see diagram below). The Think Time shall be approximated by a Delay, taken independently from the same truncated negative exponential distribution. Computing overhead for Delay initiation and completion in the RTE has to be kept to a minimum so that difference between the Delay and the effective Think Time is minimized. The maximum value of the Delay distribution must be at least 10 times the mean. The mean must be disclosed by the test sponsor.

8.6.4 Driver System Doing More than RTE Functions

In the event that a Driver System must be used to emulate additional functionality other than that described in Clause 8.4, then this must be justified as follows:

8.6.4.1 It must be demonstrated that the architecture of the proposed solution makes it uneconomical to perform the benchmark without performing the work in question on the driver. (E.g. in a client/server database implementation where the client software would run on a large number of workstations).

8.6.4.2 Clause 8.6.1 must NOT be violated.

8.6.4.3 It must be demonstrated that executables placed on the Driver System are functionally equivalent to those on the proposed (target) system.

8.6.4.4 It must be demonstrated that performance results are not enhanced by performing the work in question on the Driver System. It is the intent that a test should be run to demonstrate that the functionality, performance and connectivity of the emulated solution is the same as that for the priced system.

8.6.4.5 Individual contexts must continue to be maintained from the RTE through to the SUT.

8.6.4.6 A complete functional diagram of both the benchmark configuration and the configuration of the proposed (target) system must be provided. A detailed list of all software and hardware functionality being performed on the Driver System, and its interface to the SUT, must be provided.

8.6.5 Disclosure of Network Configuration and Emulated Portions

The test sponsor shall describe completely the network configurations of both the tested system and the proposed real (target) system which is being represented. A thorough explanation of exactly which parts of the proposed configuration are being replaced by the driver system must be given.

8.6.6 Limits on Concentration

The level of concentration of messages between the Driver System and the SUT in the benchmark configuration must not exceed that which would occur in the proposed (target) configuration. In particular, the number of communications packets which can be concentrated must not exceed the number of terminals which would be directly connected to that concentrator in the proposed configuration.

Comment: The intent is to allow only first level concentration on the RTE, but does not preclude additional levels of concentration on the SUT.
8.7 Reporting Metrics

A differentiation must be maintained between reporting of results in wide area and local attach methods of benchmarking. For reporting the throughput of the systems in units of transactions per second, the terminology should be “tpsA-Local” for the local attach method, and “tpsA-Wide” for the wide area approach. These two metrics are NOT comparable with each other.
CLAUSE 9: Pricing

9.1 Pricing Methodology

9.1.1 The intent of this section is to define the methodology to be used in calculating the price/tps. The fundamental premise is that what is tested and/or emulated is priced and what is priced is tested and/or emulated.

9.1.2 The proposed system to be priced is the aggregation of the SUT, terminals and network components that would be offered to achieve the reported performance level. Calculation of the priced system consists of:

- Price of the SUT as tested and defined in Clause 8.3;
- Price of the emulated terminals and network proposed components;
- Price of on-line storage for 90 days of history records;
- Price of additional products that are required for the operation, administration or maintenance of the priced system;
- Price of additional products required for application development.

9.1.3 Pricing methodology assumptions:

- All hardware and software used in the calculations must be announced and generally orderable by customers, with a full disclosure of the committed delivery date for general availability of products not already generally released;
- Generally available discounts for the priced configuration are permissible;
- Generally available packaged pricing is acceptable;
- Local retail pricing and discount structure should be used in each country for which results are published;
- Price should be represented by the currency with which the customer would purchase the system.

Comment 1: The intent of the pricing methodology is to allow packaging and pricing that is generally available to customers, and to explicitly exclude promotional and/or limited availability offerings.

Comment 2: Revenue discounts based on total price are permissible. Any discount must be only for the configuration being priced and can not be based on past or future purchases; individually negotiated discounts are not permitted; special customer discounts (e.g. GSA schedule, educational schedule) are not permitted.

Comment 3: The intent is to benchmark the actual system which the customer would purchase. However, it is realized that typically, vendors will announce new products and disclose benchmark results before the products have actually shipped. This is allowed, but it specifically excludes any use of "one of a kind" hardware/software configurations which the vendor does not intend to ship in the future. Products must be generally available in the country where the SUT is priced.

9.2 Priced System

9.2.1 SUT

The entire price of the SUT as configured during the test must be used, including all hardware (new purchase price), software (license charges) and hardware/software maintenance charges over a period of 5 years (60 months). In the case where the driver system provides functionality in addition to the RTE described in Clause 8.4.2, then the price of the emulated hardware/software described in Clause 9.2.2.1 are to be included.

Comment 1: The intent is to price the tested system at the full price a customer would pay. Specifically prohibited are the assumption of other purchases, other sites with similar systems, or any other assumption which relies on the principle that the customer has made any other purchase from the vendor. This is a one time, stand-alone purchase.

9.2.2 Terminals and Network Pricing
9.2.2.1 The price of the driver system is not included in the calculation, although the price of the devices the
driver is emulating (controllers, multiplexors, systems used as concentrators, LAN components, front-end
processors, workstations and terminals are some examples) are to be included.

9.2.2.2 The terminals must be commercially available products capable of entering via a keyboard all alphabetic
and numeric characters and capable of displaying simultaneously the data and the fields described in Clause
1.3.2.

9.2.2.3 LAN Pricing - For the purposes of pricing, all components from the terminal to the SUT excluding LAN
or direct connect cables must be priced.

9.2.2.4 WAN Pricing - For the purposes of pricing, the number of terminals to be connected to a single 64 kbps
(or less) line must be no greater than that emulated per Clause 8.5.1.2. All hardware components which are
required to connect to the 64 kbps line must be included in the pricing. The price of the 64 kbps line(s) is
excluded.

Comment: The intent is that all components including PADS (packet assemblers-disassemblers), modems,
concentrators, multiplexors, etc. required to attach to the 64 kbps line must be priced in addition to the price of
the terminals/ workstations.

9.2.3 History Storage and Recovery Log Pricing

9.2.3.1 Within the priced system, there must be sufficient on-line storage to support 8 hours of recovery log
data plus any other expanding system files (see Clause 7.1) and durable history records/ rows for 90 eight-hour
days at the published tps rate, i.e. 90 x 8 x 60 x 60 = 2,592,000 records/ rows per tps. On-line storage includes
magnetic disks, magnetic tapes, optical disks, and any combination of these. Storage is considered on-line if any
record can be accessed randomly within one second.

Comment: The 90-day history file is required so as to force configuration of a realistic amount of on-line storage.

9.2.3.2 For purposes of pricing storage for history records/ rows, any unused on -line storage present in the SUT
may count towards the history storage requirements. (However, note that unused storage may also be required
to satisfy the 8-hour log requirement of Clause 7.1).

9.2.3.3 If it is necessary to price any additional storage devices for history data, such devices must be of the
type(s) actually used in the SUT during the test, and must satisfy the normal system configuration rules.

Comment: The intent is to exclude unrealistic on-line storage devices or configurations from the pricing
procedure.

9.2.4 Additional Operational Components

9.2.4.1 Additional products that might be included on a customer installed configuration, such as operator
consoles, magnetic tape drives and printers, are also to be included in the priced system if explicitly required for
the operation, administration or maintenance of the priced system.

9.2.4.2 Copies of the software on appropriate media, and a software load device if required for initial load or
maintenance updates, must be included.

9.2.4.3 The price of an Uninterruptible Power Supply specifically contributing to a durability solution must be
included (see Clause 2.5.3.2).

9.2.5 Additional Software

9.2.5.1 The price must include the software licenses necessary to create, compile, link and execute this
benchmark application; as well as all run-time licenses required to execute on host system(s) and connected
workstations.

9.2.5.2 In the event the application code is developed on a system other than the SUT, the price of that system
and any compilers and other software used must also be included as part of the priced system.

9.3 Maintenance
9.3.1 Hardware and software maintenance must be figured at a standard pricing which covers at least 5 days/week, 8 hours/day coverage, either on-site, or if available as standard offering, via a central support facility. Hardware maintenance maximum response time must not exceed 4 hours, on any part whose replacement is necessary for the resumption of operation.

9.3.2 If central support is claimed, then the appropriate connection device, such as auto-dial modem must be included in the hardware price. Also any software required to run the connection to the central support, as well as any diagnostic software which the central support facility requires to be resident on the tested system, must not only be included in pricing, but must also be installed during the benchmark runs.

9.3.3 Software maintenance must include update distribution for both the software and documentation. If software maintenance updates are separately priced, then pricing must include at least 3 updates over the 5 year period.

**Exception:** Maintenance and warranty terms for terminals and workstations must cover at a minimum a return for repair service.
CLAUSE 10: Full Disclosure

10.1 Full Disclosure Report Requirements

A full disclosure report is required for results to be considered compliant with TPC Benchmark A specifications.

Comment: The intent of this disclosure is for a customer to be able to replicate the results of this benchmark given the appropriate documentation and products.

A full disclosure report must include the following:

10.1.1 General Items

10.1.1.1 A statement identifying the sponsor of the benchmark and any other companies who have participated.

10.1.1.2 Program listing of application code and definition language statements for files/ tables.

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

- Database options;
- Recovery/commit options;
- Consistency/locking options;
- System parameters, application parameters, and configuration parameters.

Test sponsors may optionally provide a full list of all parameters and options.

10.1.1.4 Configuration diagrams of both benchmark configuration and the priced system, and a description of the differences.

10.1.2 Clause 2 Related Items

10.1.2.1 Results of the ACIDity test (specified in Clause 2) must describe how the requirements were met.

10.1.3 Clause 3 Related Items

10.1.3.1 The distribution across storage media of ABTH (Accounts, Branch, Teller, and History) files/ tables must be explicitly depicted.

10.1.3.2 A description of how the database was populated, along with sample contents of each ABTH files/ tables to meet the requirements described in Clause 3.

10.1.3.3 A statement of the type of database utilized, e.g., relational, Codasyl, flat file, etc.

10.1.4 Clause 5 Related Items

10.1.4.1 The method of verification of the random number generator should be described.

10.1.5 Clause 6 Related Items

10.1.5.1 Both maximum and average response time shall be reported, as well as performance curves for tps vs. response time and response time distribution (see Clauses 6.6.1 and 6.6.2).

10.1.6 Clause 7 Related Items

10.1.6.1 The method used to determine that the SUT had reached a steady state prior to commencing the measurement interval should be described.

10.1.6.2 A description of how the work normally performed during a sustained test (for example checkpointing, writing redo/undo log records, etc., as required by Clause 7.2), actually occurred during the measurement interval.

10.1.6.3 A description of the method used to determine the reproducibility of the measurement results.
10.1.6.4 A statement of the duration of the measurement period for the reported tps (it should be at least 15 minutes).

10.1.7 Clause 8 Related Items

10.1.7.1 If the RTE is commercially available, then its inputs should be specified. Otherwise, a description should be supplied of what inputs (e.g., scripts) to the RTE were used.

10.1.7.2 A proof that the functionality and performance of the components being emulated in the Driver System are equivalent to that of the priced system.

10.1.7.3 If the SUT contains a WAN or a LAN network, its bandwidth should be specified.

10.1.8 Clause 9 Related Items

10.1.8.1 A detailed list of hardware and software used in the priced system. 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.

10.1.8.2 The total price of the entire configuration is required including: hardware, software and maintenance charges. Separate component pricing is recommended. The basis of all discounts used shall be disclosed.

10.1.8.3 A statement of the measured tpsA-Wide or tpsA-Local, and the calculated price/tpsA-Wide or price/tpsA-Local.

10.1.9 Clause 11 Related Items

10.1.9.1 If the benchmark has been independently audited, then the auditor’s name, address, phone number, and a brief audit summary report indicating compliance must be included in the full disclosure report. A statement should be included, specifying when the complete audit report will become available and who to contact in order to obtain a copy.

10.2 Availability of the Full Disclosure Report

The full disclosure report is to be readily available to the public at a reasonable charge, similar to charges for similar documents by that test sponsor. The report is to be made available when results are made public. In order to use the phrase "TPC Benchmark™ A", the full disclosure report must have been submitted to the TPC Administrator as well as written permission to distribute same.

10.3 Revisions to the Full Disclosure Report

Revisions to the full disclosure documentation shall be handled as follows:

10.3.1 Fully documented price changes can be reflected in a new published price/throughput. The benchmark need not be rerun to remain compliant.

10.3.2 Product substitutions within the SUT require the benchmark to be re-run with the new components in order to re-establish compliance.

10.3.3 The revised report should be submitted as defined in 10.2.

Comment: During the normal product life cycle problems will be uncovered which require changes, sometimes referred to as ECOs, FCOs, Patches, Updates, etc. If any of these changes causes the tps rating of the system to change by more than 5%, then the test sponsor will be required to re-validate the benchmark results.

10.4 Official Language

10.4.1 The official full-disclosure report must be written in English but may be translated to additional languages.
CLAUSE 11: Audit

11.1 An independent audit of the benchmark results is highly recommended. An audit checklist is provided as part of this specification.

11.2 The audit report is to be made readily available to the public at a reasonable charge, similar to charges for similar documents.

11.3 Auditor's check list:

11.3.1 Clause 1 Related Items

11.3.1.1 Verify that the application program matches the transaction profile of Clause 1.2.

11.3.1.2 Verify that message sizes and content satisfy Clause 1.3 and that message compression is not used.

11.3.2 Clause 2 Related Items

11.3.2.1 Verify that the requirements of each of the ACIDity tests were met.

11.3.3 Clause 3 Related Items

11.3.3.1 For each of the ABTH files verify that specified fields/columns and records/rows exist, and that they conform to the minimum lengths specified in Clause 3.2.

11.3.3.2 Verify that the ABT record/row identifiers are not disk or file offsets as specified in Clause 3.4.

11.3.3.3 Verify that the ABT files/tables support retrievals, inserts and deletes as specified in Clause 3.5.

11.3.4 Clause 4 Related Items

11.3.4.1 Verify that the ratios among the numbers of records/rows of each file/table are as specified in Clause 4.2.

11.3.4.2 Verify that the total number of tellers is at least 10 times the system's tps rating as specified in Clause 4.2.1.

11.3.4.3 Verify randomness of the Account_ID, Branch_ID, and Teller_ID sequences input of the system under test. Include verification that the values generated are uniform across the entire set of accounts necessary to support the claimed tps rating per Clause 4.4 (scaling).

11.3.5 Clause 5 Related Items

11.3.5.1 Verify that at least 15% of the transactions are remote, and that the distribution of Account_IDs of remote transaction is uniform across non-home branches.

11.3.5.2 If horizontal partitioning is used, establish whether or not it is transparent to the application program.

11.3.5.3 Verify that vertical partitioning of the ABTH files is not used.

11.3.6 Clause 6 Related Items

11.3.6.1 Verify the validity of the method used to measure the response time at the RTE.

11.3.6.2 If part of the SUT is emulated, verify that the reported response time is no less than the response time that would be seen by a real terminal user.

11.3.7 Clause 7 Related Items

11.3.7.1 Verify the method used to determine that the SUT had reached a steady state prior to commencing the measurement interval.
11.3.7.2 Verify that all work normally done in a steady state environment actually occurred during the measurement interval, for example checkpointing, writing redo/undo log records to disk, etc., per Clause 7.2, Comment 2.

11.3.7.3 Verify the method used to determine the reproducibility of the measurement results.

11.3.7.4 Verify the duration of the measurement period for the reported tps (at least 15 minutes).

11.3.7.5 Verify that the response time and the tps were measured in the same time interval.

11.3.8 Clause 8 Related Items

11.3.8.1 Describe the method used to verify the accurate emulation of the tested terminal community by the driver system if one was used.

11.3.9 Clause 9 Related Items

11.3.9.1 Verify that all application development software is installed on the priced system and has been used to compile, link and execute the benchmark.

11.3.9.2 Verify that pricing includes all the hardware and software licenses as required in Clause 9.

11.3.9.3 Verify that the priced configuration includes sufficient disk storage for the database, 8 hours of logs and 90 days of history records at the measured tps rate and can be configured in the priced system.

11.3.9.4 Assure that warranty coverage meets the requirements of Clause 9.3, or that additional costs for maintenance have been added to priced system.

11.3.9.5 Verify that all prices used, including discounts, are generally available.
APPENDIX A: Sample Implementation

/*
* This is a sample implementation of the Transaction Processing Performance
* Council Benchmark A coded in ANSI C and ANSI SQL2.
* Any equivalent implementation is equally acceptable.
*
* Exceptions:
* 1. Since no standard syntax exists for networking, C standard IO is used.
*    In an actual benchmark, this must be replaced with WAN or LAN
*    message software.
* 2. ANSI/ISO SQL have no explicit BEGIN WORK (begin transaction).
*    To show that message handling is outside the transaction,
*    explicit BEGIN WORK statements are included
* 3. The C language has only integer and float numerics - it does not
*    support precision or scale. So, in this implementation, money is
*    represented as integer pennies (pence, pfennig, centimes,...)
* 4. To clarify the schema, the following SQL2 features are used:
*    Primary Key
*    Foreign Key
*    DateTime datatype
*    Default values (to simplify handling of pad chars).
* 5. For simplicity, the program does no error checking or handling.
*/

/* Global declarations */
exec sql BEGIN DECLARE SECTION;

/* tpc bm a scaling rules */
long tps = 1;  /* the tps scaling factor: here it is 1*/
long nbranches = 1;  /* number of branches in 1 tps db */
long ntellers = 10;  /* number of tellers in 1 tps db */
long naccounts = 100000;  /* number of accounts in 1 tps db */
long nhistory = 2592000;  /* number of history recs in 1 tps db */

/* working storage */
long i,sqlcode, Bid, Tid, Aid, delta, Abalance;

exec sql END DECLARE SECTION;

void CreateDatabase();
long DoOne();
#include <stdio.h>

/* main program, */
*
* Creates a 1-tps database, ie 1 branch, 10 tellers,...
* runs one TPC BM A transaction
*/
main()
{
    CreateDatabase();
    DoOne();
}
/* CreateDatabase - Creates and Initializes a scaled database. */

void CreateDatabase()
{

exec sql BEGIN WORK; /* start trans to cover DDL ops */
exec sql CREATE TABLE branches ( 
  Bid        NUMERIC(9), PRIMARY KEY(Bid),
  Bbalance   NUMERIC(9),
  filler     CHAR(92) DEFAULT SYSTEM
); /* pad to 100 bytes */
exec sql CREATE TABLE tellers ( 
  Tid        NUMERIC(9), PRIMARY KEY(Tid),
  Bid        NUMERIC(9) FOREIGN KEY REFERENCE TO branches,
  Tbalance   NUMERIC(9),
  filler     CHAR(88) DEFAULT SYSTEM
); /* pad to 100 bytes */
exec sql CREATE TABLE accounts ( 
  Aid        NUMERIC(9), PRIMARY KEY(Aid),
  Bid        NUMERIC(9) FOREIGN KEY REFERENCE TO branches,
  Abalance   NUMERIC(9),
  filler     CHAR(88) DEFAULT SYSTEM
); /* pad to 100 bytes */
exec sql CREATE TABLE history ( 
  Tid        NUMERIC(9) FOREIGN KEY REFERENCE TO tellers,
  Bid        NUMERIC(9) FOREIGN KEY REFERENCE TO branches,
  Aid        NUMERIC(9) FOREIGN KEY REFERENCE TO accounts,
  delta      NUMERIC(9),
  time       TIMESTAMP,
  filler     CHAR(26) DEFAULT SYSTEM
); /* pad to 50 bytes */

/* prime database using TPC BM A scaling rules.
Note that for each branch and teller:
branch_id = teller_id / ntellers
branch_id = account_id / naccounts */
for (i = 0; i < nbranches*tps; i++)
  exec sql INSERT INTO branches(Bid,Bbalance) VALUES (:i,0);
for (i = 0; i < ntellers*tps; i++)
  exec sql INSERT INTO tellers(Tid,Bid,Tbalance) VALUES (:i,:i/:ntellers,0);
for (i = 0; i < naccounts*tps; i++)
  exec sql INSERT INTO accounts(Aid,Bid,Abalance) VALUES (:i,:i/:naccounts,0);
exec sql COMMIT WORK;
} /* end of CreateDatabase */
/* 
  * DoOne - Executes a single TPC BM A transaction. 
  */

void DoOne()
{
    scanf("%ld %ld %ld %ld", &Bid, &Tid, &Aid, &delta); /* note: must pad to 100 bytes*/

    exec sql BEGIN WORK;

    exec sql UPDATE accounts
        SET      Abalance = Abalance + :delta
        WHERE    Aid = :Aid;

    exec sql SELECT Abalance INTO :Abalance
        FROM      accounts
        WHERE     Aid = :Aid;

    exec sql UPDATE tellers
        SET      Tbalance = Tbalance + :delta
        WHERE    Tid = :Tid;

    exec sql UPDATE branches
        SET      Bbalance = Bbalance + :delta
        WHERE    Bid = :Bid;

    exec sql INSERT INTO history(Tid, Bid, Aid, delta, time)
        VALUES (:Tid, :Bid, :Aid, :delta, CURRENT);

    exec sql COMMIT WORK;

    printf("%ld, %ld, %ld, %ld\n", Bid, Tid, Aid, Abalance, delta);
        /* note: must pad to 200 bytes */
}
    /* end of DoOne */