TPC BENCHMARK™ A

Standard Specification
Revision 2.0

7 June 1994

Transaction Processing Performance Council (TPC)

Administered by
Shanley Public Relations
777 North First St., Suite 600
San Jose, CA 95112, USA
Phone: (408) 295-8894
FAX: (408) 295-2613
e-mail: shanley@cup.portal.com

TPC MEMBERSHIP
(June 1994)

Amdahl	IDEAS International	Samsung Electronics
AST Research	Informix	SCO
AT&T/NCR/Teradata	INGRES	Sequent Computer
Australian Government	Intel Corp.	Siemens Nixdorf
Bull S.A.	Intergraph	Silicon Graphics
Compaq	ITOM International	Software AG
Convex Computer	KPMG Peat Marwick	Solbourne
Cray Research	Microsoft	Sony
Data General	Mitsubishi	Stratus Computer
Digital Equipment Corp.	NEC	Sun Microsystems
EDS	Novell	Sybase
Encore	OKI Electric Industry	Tandem Computers
Fujitsu/ICL	Olivetti	Toshiba
Hewlett Packard	Oracle	Tricord Systems
Hitachi Ltd.	Performance Metrics	Unisys
IBM	Pyramid Technology	

Document History:

Date	Version	Description
10 November 1989	First Edition	Standard specification released to public.
1 March 1992	Revision 1.1	Revised standard specification.
16 March 1993	Revision 1.2	Second revision.
7 June 1994	Revision 2.0	Third revision.

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

TABLE OF CONTENTS

0.1   Introduction

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 benchmark uses terminology and metrics which are similar to other benchmarks, originated by the TPC and 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. Comparison of price/performance results disclosed in one country may not be meaningful in another country because of pricing and product differences.

The throughput metric is "tpsA". TPC Benchmark™ A can be run in a wide area or local area network configuration. No distinction is made between results for wide area and local area configurations.

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 reported 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.

> While separated from the main text for readability, "comments" are a part of the standard and must be enforced. The sample implementation included as Appendix A is provided only as an example and is specifically not part of the standard.

par

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

- Are generally available to users.

- Are relevant to the market segment that the individual TPC benchmark models or represents (e.g. TPC-A models and represents high-volume, simple OLTP environments).

- A significant number of users in the market segment the benchmark models or represents would plausibly implement.

The use of new systems, products, technologies (hardware or software) and pricing is encouraged so long as they meet the requirements above. Specifically prohibited are benchmark systems, products, technologies, pricing (hereafter referred to as "implementations") whose primary purpose is performance optimization of TPC benchmark results without any corresponding applicability to real-world applications and environments. In other words, all "benchmark specials," implementations that improve benchmark results but not real-world performance or pricing, are prohibited.

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

The following characteristics should be used to judge whether a particular implementation is a benchmark special:

- Is the implementation generally available, documented, and supported?
- Does the implementation have significant restrictions on its use or applicability that limits its use beyond TPC benchmarks?
- Is the implementation or part of the implementation poorly integrated into the larger product?
- Does the implementation take special advantage of the limited nature of TPC benchmarks (e.g., transaction profile, transaction mix, transaction concurrency and/or contention, transaction isolation) in a manner that would not be generally applicable to the environment the benchmark represents?
- Is the use of the implementation discouraged by the vendor? (This includes failing to promote the implementation in a manner similar to other products and technologies.)
- Does the implementation require uncommon sophistication on the part of the end-user, programmer, or system administrator?
- Is the pricing unusual or non-customary for the vendor or unusual or non-customary to normal business practices? The following pricing practices are suspect:
  - Availablility of a discount to a small subset of possible customers.
  - Discounts documented in an unusual or non-customary manner.
  - Discounts that exceeds 25% on small quantities and 50% on large quantities.
  - Pricing featured as a close-out or one-time special.
  - Unusual or non-customary restrictions on transferability of product, warranty or maintenance on discounted items.
- Is the implementation being used (including beta) or purchased by end-users in the market area the benchmark represents? How many? Multiple sites? If the implementation is not currently being used by end-users, is there any evidence to indicate that it will be used by a significant number of users?

1.1 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 Clause 1.2.ar1.1.2 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 andlogical data structures.

~~If the application environment contains software that routes or organizes the execution of transactions (e.g., a transaction processing monitor), the software must be a generally available, commercial product that is fully supported as defined in Clause 9.~~

~~**Comment:** It is the intent that special purpose transaction processing monitors developed specifically for benchmarking or limited use not be utilized.~~

1.1.3 If transactions are routed or organized within the SUT, a commercially available transaction processing monitor or equivalent commercially available software (hereinafter referred to as TM) is required with the following features/functionality:

Operation - The TM must allow for:
- request/service prioritization
- multiplexing/de multiplexing of requests/services
- automatic load balancing
- reception, queuing, and execution of multiple requests/services concurrently

Security - The TM must allow for:
- the ability to validate and authorize execution of each service at the time the service is requested.
- the restriction of administrative functions to authorized users.

Administration/Maintenance - The TM must have the pre-defined capability to perform centralized, non programmatic (i.e., must be implemented in the standard product and not require programming) and dynamic configuration management of TM resources including hardware, network, services (single or group), queue management prioritization rules, etc.

Recovery - The TM must have the capability to:
- post error codes to an application.
- detect and terminate long-running transactions based on predefined time-out intervals.

Application Transparency - The message context(s) that exist between the client and server application programs must be managed solely by the TM. The client andserver application programs must not have any knowledge of the message context or the underlying communication mechanisms that support that context.

Comment 1: The following are examples of implementations that are non-compliant with the Application Transparency requirement:

1. Client and server application programs use the same identifier (e.g., handle or pointer) to maintain the message context for multiple transactions.
2. Change and/or recompilation of the client and/or server application programs is required when the number of queues or equivalent data structures used by the TM to maintain the message context between the client and server application programs is changed by TM administration.

Comment 2: The intent of this clause is to exclude special purpose software developed for benchmarking or other limited use.

Comment 3: Functionality of TM or equivalent software is not required if the DBMS maintains an individual context for each emulated user.

~~1.1.3~~ 1.1.4   Implementors of this benchmark are permitted many possible system designs, insofar as they adhere to the standard model described and illustrated in Clause 8.

~~1.1.4~~ 1.1.5   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 Clause 9.2.2.2).

1.2   **The Transaction Profile**

Read 100 bytes including Aid, Tid, Bid, Delta from terminal (see Clause 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 Clause 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 Inputs and Outputs**

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 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.

1.4.3   There is no requirement for a separate, explicit read from the Account table to return the account balance.

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.

These tests are intended to demonstrate the ACID properties are supported by the system under test and enabled during the performance measurement period. The tests are not intended to be an exhaustive quality assurance test.

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, including remote transactions that are processed on two nodes, satisfy the ACID properties. (See Clause 5 for the definition of home and remote transactions.)

2.1.4 Test sponsors reporting TPC results may perform ACID tests on any one system for which results have been disclosed provided that they used the same software executables (e.g., OS, database, transaction, etc.). For example, this would be applicable when results are reported for multiple systems in a product line. However, the durability tests described in Clauses 2.5.3.2 and 2.5.3.3 must be run on all systems that are measured. All disclosure reports must identify the systems which were used to verify ACID requirements and full details of the ACID tests conducted and results obtained.

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 (Clauses 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 tpsA (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 Clause 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 Clause 2.3.3.2, Step 2, plus the number of transactions reported as committed in Clause 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 Clauses 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 Clause 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, memory can be considered a durable medium if it is accompanied by an uninterruptible power supply and the contents 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 Clauses 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 required that the system crash test and the loss of memory test described in Clauses 2.5.3.2 and 2.5.3.3, respectively, be performed with a full terminal load and a fully scaled database. The durable media failure tests described in Clause 2.5.3.1 may be performed on a subset of the SUT configuration and database. For that subset, all multiple hardware components, such as processors and disks/controllers in the full configuration must be represented by either 10% or 2 each of themultiple hardware components, whichever is greater.   The database subset must be scaled to at least 10% (minimum of 2 tps) of the fully scaled database size. The test sponsor must state that to the best of their knowledge, a fully loadedand fully scaledtest SUT and database configuration would also pass all durability tests.

At the time of the induced failures, it is required to have multiple home and remote transactions (see Clause 5) in progress. Distributed configurations must have distributed transactions in progress as well.

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

Step 1:   Perform Step 1 of the History file Consistency Test in Clause 2.3.3.3.

Step 2:   Start submitting TPC Benchmark™ A transactions. On the driver system, record committed transactions in a "success" file.

Step 3:   Cause a failure selected from the list in Clause 2.5.3.

Step 4:   Restart the system under test using normal recovery procedures.

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).

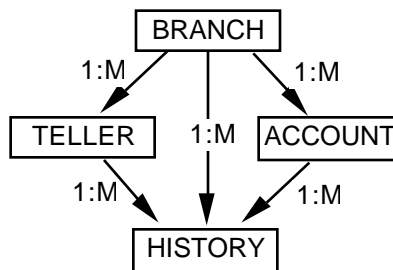Step 6: Perform the consistency test on the Branch and Teller files as specified in Clause 2.3.3.2.

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 relationshipsamong these files/tables are defined in the following entity/relationship diagram and are subject to the business rules specified in Clause 3.1.2. This diagram is a logical description and has no implication for physical implementation.ar**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.

```
                        ┌──────────┐
                        │  BRANCH  │
                        └──────────┘
                  1:M   /    │    \   1:M
                       /     │     \
               ┌──────────┐ 1:M ┌──────────┐
               │  TELLER  │     │ ACCOUNT  │
               └──────────┘     └──────────┘
                  1:M   \   │   /   1:M
                         \  │  /
                        ┌──────────┐
                        │ HISTORY  │
                        └──────────┘
```

3.1.2   The entities in Clause 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.

Comment: There is no intent to model an environment in which accounts and tellers can be moved from branch to branch.

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 handledby 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 Clauses 3.2.2 through 3.2.5. Any space with unspecified values in the record/row descriptions in Clauses 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 be 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.r3.3
        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 tpsA 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 values.

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 numberssince 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.

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):

Account records/rows	100,000
Teller records/rows	10
Branch records/rows	1
History record/rows	(See Clause 4.3)
Terminals	10

4.2.1   All terminals should be active throughout the steady state period. The intent is that each terminal should contribute no more than 1/10th 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.ar4.2.2Should   any value in Clause 4.2 be exceeded, the others should be increased proportionately to maintain the same ratios among them as in Clause 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-tpsA, storage must be maintained for the number of history records specifiedin 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 tpsA may not exceed the configured (nominal) rate represented by the file/table sizes in Clause 4.2. While the reported tpsA may fall short of the maximum allowed by the configured system, the price-per-tpsA computation must report the priceof the system as actually configured.

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, detailsof the partitioning and transaction program logic 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 or branches = 1, a random Account_ID is selected over all <Branch_ID> accounts.
- If X>=0.85 and branches > 1, a random Account_ID is selected over all non-<Branch_ID> accounts.

Comment 1: This algorithm guarantees that, if there is more than one branch in the database, then 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 ofremote transactions may vary around 15%.   Actual measured values must be within 14% to 16% for the set of transactions processed during the measurement interval (see Clauses 6.1 and 7.2).

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 dividedevenly 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.

5.3.5   All transactions during steady state should be uniformly distributed over all Teller_ID's, within normal statistical variations.

5.4 Definition of "Random"

Within Clause 5, the term random means independently selected and uniformly distributed.

6.1 Measurement Interval and Timing

6.1.1   In this clause, the term "measurement interval" isthe steady state period (see Clause 7.1) during the execution of the benchmark for which the test sponsor is reporting a tpsA 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.

The resolution of the timestamps must be at least 0.1 seconds.

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 criterion has been chosen to provide a single criterion 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 tpsA 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   For reporting the throughput of the SUT in units of transactions per second, the terminology is "tpsA".

6.4.3   Reported tpsA must be expressed to exactly two decimal places, rounded to the hundredth place.

6.5 Interpolation and Extrapolation Prohibited

The reported tpsA rate must be measured rather than interpolated or extrapolated. For example, suppose 9.13 tpsA is measured on a 100 terminal test during which 90% of the transactions completed in less than 1.7 seconds and 9.77 tpsA is measured on a 110 terminal test during which 90% of the transactions completed in less than 2.3 seconds. Then the reported tpsA is 9.13 rather than some interpolated value between 9.13 and 9.77.

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, ofequal one-second-or-less length, must be reported. A sample graph is shown below. The maximum and average response times must also be reported.

Average Response Time

90th Percentile
Response Time

Number of Transactions

Response time (seconds)

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
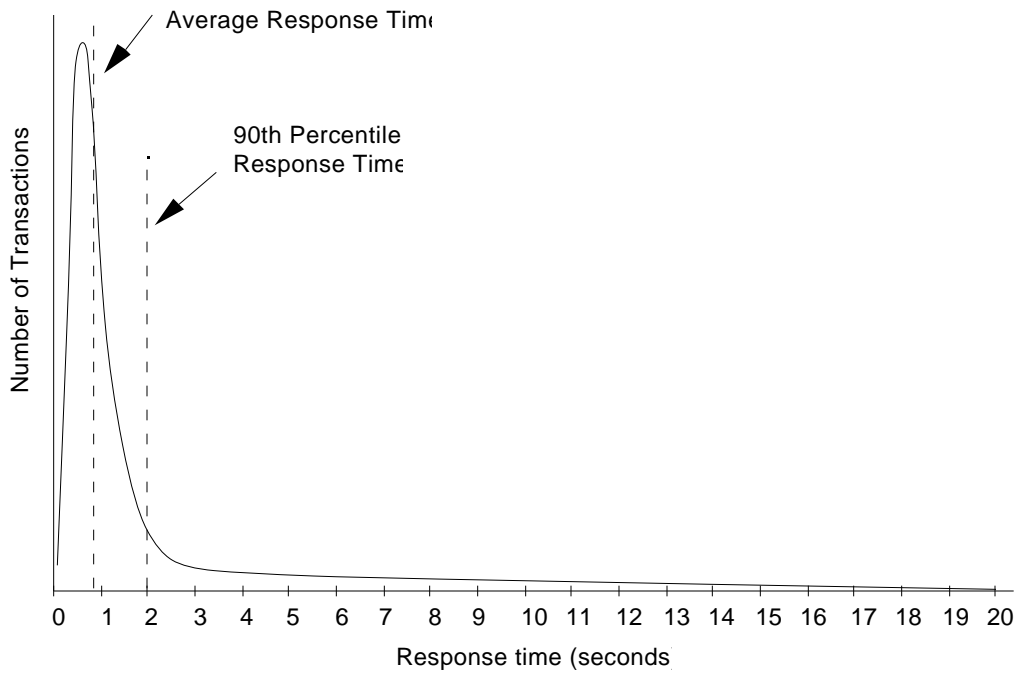
6.6.2   A complete curve of response times versus tpsA must be reported. The points on the curve must be of the form (x,y), where:

x = measured tpsA

y = 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.
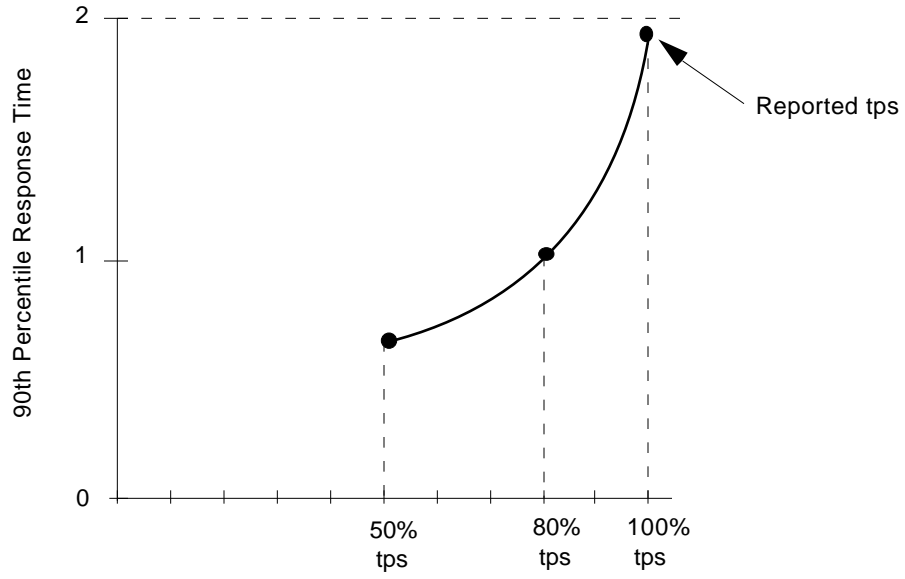
7.1 Steady State

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 tpsA for a continuous period of at least eight hours, maintaining full ACID properties. For example, the media used to store at least eight hours of log data must be configured, if required to recover from any single point of failure (see Clause 2.5.3.1).

Comment: An example of a configuration that would not comply is one where a log file is allocated such that better performance is achieved during the measured portion of the test than during the remaining portion of an eight hour test, perhaps because a dedicated device was used initially but space on a shared device is used later in the full eight hour test.

7.2 Duration and Requirements

The measurement period must:

- Begin after the system reaches sustained "steady state";
- Be long enough to generate reproducible tpsA 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."

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 Clause8.3 and 8.4) where the response time is measured.
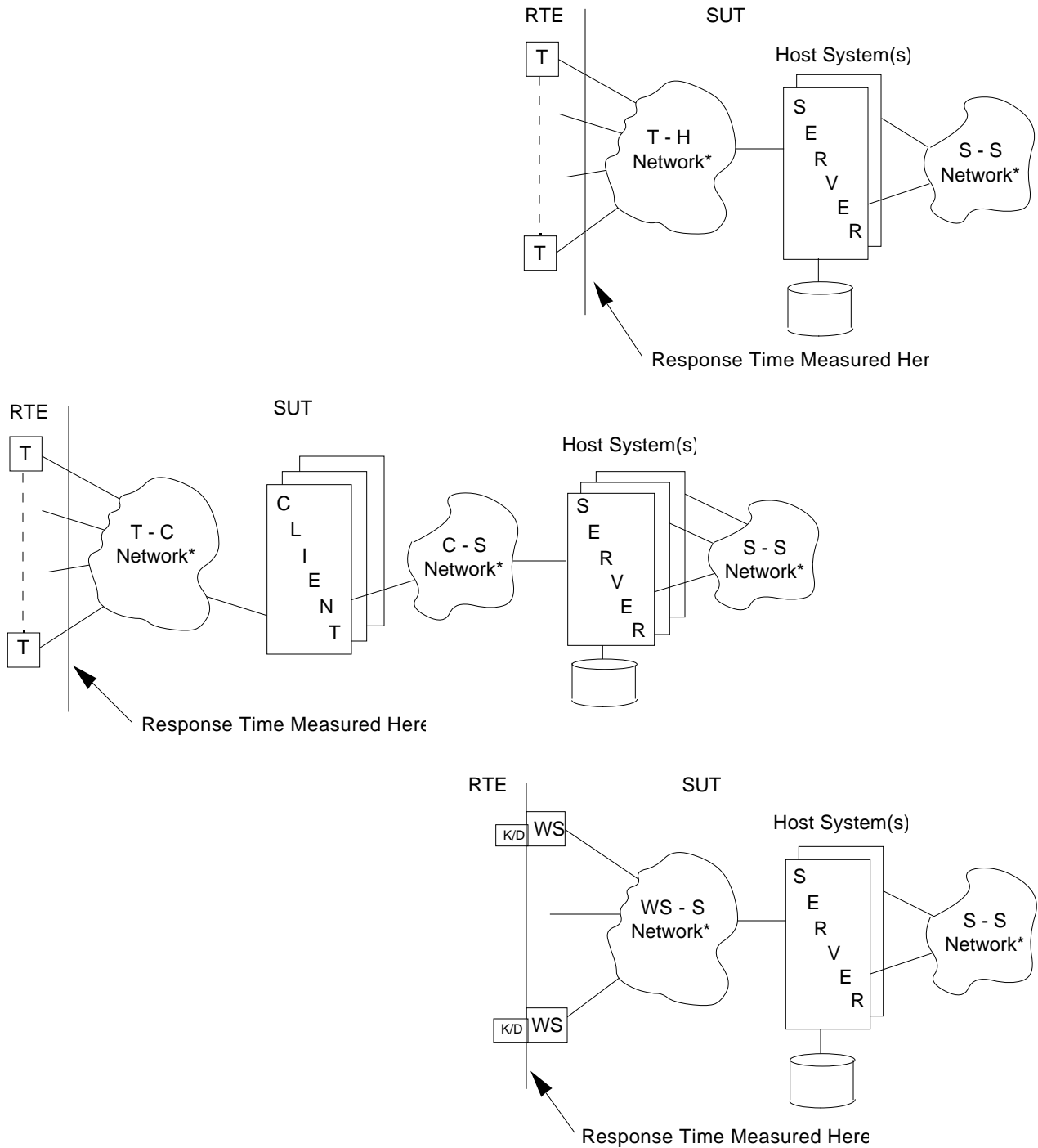
LEGEND: C - Client; H - Host; K/D - Keyboard/Display; RTE - Remote Terminal Emulator; S -Server; SUT - System Under Test; T - Terminal; WS - Workstation. * - optional.

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 aLAN-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

8.3.1   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.
- Any front-end systems areconsidered 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.

8.3.2   A single benchmark result may be used for multiple SUTs provided the following conditions are met:

- Each SUT must have the same hardware and software architecture and configuration. The only exception allowed is for elements not involved in the processing logic of the SUT (e.g., number of peripheral slots, power supply, cabinetry, fans, etc.).
- Each SUT must support the priced configuration.

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 Clause 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. (SeeClause 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.
- Configurations where the processor(s) in the SUT is 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 equallydisallowed.

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 Clause 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 isto allow pseudo-conversational transactions. The intent of Clause 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 the 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.
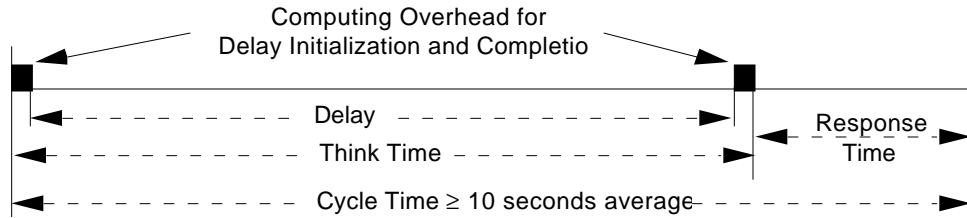
8.6.3.1 The frequency distribution of think times of transactions started and completed during the measurement interval must be reported. The range of the X-axis must be from 0 to 20 seconds think times. At least 40 different intervals, of equal 0.5 seconds or less length, must be reported. A sample graph is shown below. The maximum and average think times must also be reported.
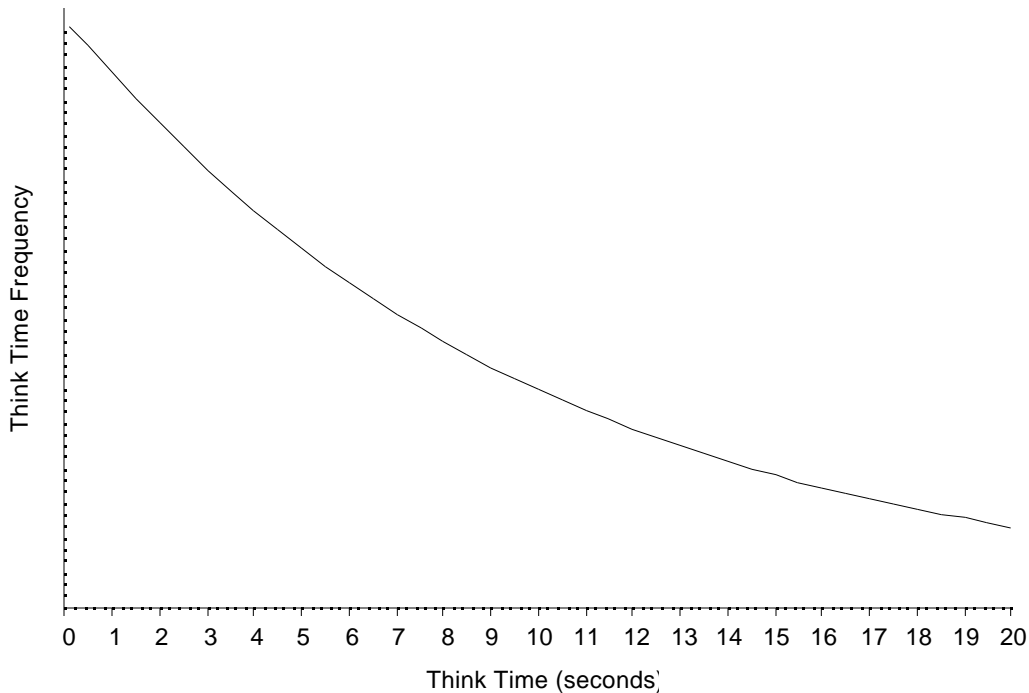
8.6.4   Driver System Doing More than RTE Functions

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 clientsoftwarewould 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.

For example, if the driver system emulated the functionality of a terminal concentrator, there must be test data to demonstrate that a real concentrator would deliver the same (or better) response time as measured in the driver system. The concentrator used in the test must be configured as it would be in the priced system, and loaded to the maximum number of lines in the priced configuration. The test must be run as a part of the SUT configuration that is running a full load on a properly scaled database. The test data must be included in the full disclosure report (FDR). The following diagram illustrates a possible example of this test:
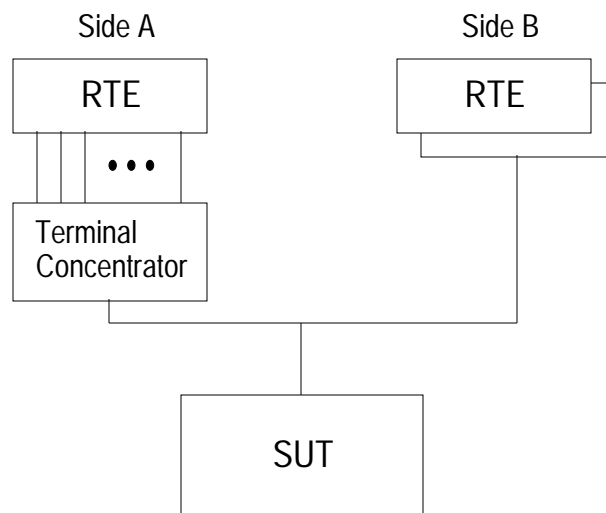
The difference in measured response time between Side A and Side B should be less than or equal to any adjustment to the response time reported in the FDR.

If the response time delay generated from this test is to be used in multiple benchmark tests, the test must be performed on a SUT generating the highest tpsA rate on the terminal concentrator.

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 whichcan 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.

9.1   Pricing Methodology

9.1.1   The intent of this section is to define the methodology to be used in calculating the price/tpsA. 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:

   a) Price of the SUT as tested and defined in Clause 8.3;
   b) Price of the emulated terminals and network proposed components;
   c) Price of on-line storage for 90 days of history records;
   d) Price of additional products that are required for the operation, administration or maintenance of the priced system;
   e) 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. For any products not already generally released, the full disclosure report (FDR) must include a committed general delivery date.That delivery date must not exceed 12 months beyond the FDR submittal date.
   • 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.
   • Currently available products must be priced using current prices on the vendor's price books and not prices that will become effective in the future.

For test sponsors who have only indirect sales channels, pricing must be actual generally available pricing from indirect channels which meet all other requirements of Clause 9.

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 cannot 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.   Thisis 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.

9.1.4   If any hardware, software, or maintenance is provided by a third party not involved as a sponsor of the benchmark, the applicable pricing must satisfy all the requirements for general availability, standard volume discounts, and full disclosure. Furthermore, any pricing which is not directly offered by the test sponsor(s) and not derived from the third party vendor's generally available pricing and discounts must be guaranteed by the third party vendor in a written quotation for a period not lessthan sixty (60) days from the date the benchmark results are submitted for review. The written quotation must be included in the full disclosure report and state that the quoted prices are generally available, the time period for which the prices are valid, the basis of all discounts offered, and any terms and conditions which may apply to the quoted prices. The test sponsor(s) must still comply with any price changes as described in Clause 8.3.

9.1.5   Pricing shown in the full disclosure report must reflect line item pricing from the vendor's price books.

Comment: The intent of this clause is that the pricing reflect the level of detail that an actual customer purchasing the priced equipment would see on an itemized billing, excluding taxes and shipping charges.

9.1.6   For publishing in another country other than the country for which the results are originally published, it is permitted to substitute local components from the original report providing the substituted products are sold to the same product description or specification.

Comment: The intention is to encourage local country pricing by allowing substitution of equipment for country specific reasons such as voltage, product numbering, industrial/safety, keyboard differences, etc., which do not affect performance.

9.1.7 Customer spareable and replaceable hardware items are acceptable under the following conditions:

   a) The spareable and replaceable hardware items must be generally available as spareable and replaceable for any customer installation.
   b) The designation as spareable and replaceable cannot depend upon a threshold of purchased quantity.
   c) It must be verifiable that a customer could successfully diagnose the failure of a spareable and replaceable item within four (4) hours of the failure.
   d) The method for diagnosis and replacement must have complete customer documentation.

Comment: Diagnosis may take the form of a hardware indicator or diagnosis procedure. The intent is that diagnosis must reach a positive conclusion as to the state of the hardware item within four (4) hours.

9.2   **Priced System**

9.2.1   <u>SUT</u>

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 thatthe customer hasmade any other purchase from the vendor. This is a one time, stand-alone purchase.

Comment 2: The number of users for TPC-A is defined to be equal to the number of terminals included in the priced configuration. Any usage pricing for the above number of users should be based on the pricing policy of the company supplying the priced component.

9.2.2   <u>Terminals and Network Pricing</u>

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 andterminals 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.ar9.2.3   History Storage and Recovery Log Pricing

9.2.3.1 Within the priced system, there must be sufficient on-line storage to support any expanding system files.

1. The amount of space required must be priced as follows: durable History records/rows at the published tps rate should be calculated based on storage used during measurement.

2. The above calculated space will be multiplied by 90.

Comment 1: The 90-day history file is required so as to force configuration of a realistic amount of on-line storage.

Comment 2: Storage is considered to be on-line if any record can be accessed randomly within one second. On-line storage media include magnetic disk, magnetic tape, optical disk, and any combination of the previous.

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 needed for expanding system files as required in Clauses 7.1 and 9.2.3.1.)

9.2.3.3 If it is necessary to price any additional storage devices to fulfill the ninety (90) day history storage requirement, such devices must be of thetype(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.

9.2.3.4 The requirement to support 8 hours of recovery log data can be met with storage on any durable media (see Clause 2.5.1) if all data required for recovery from failures listed in 2.5.3.2 and 2.5.3.3 are on-line.

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 maintenanceof 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. Resumption of operation means the priced system must be returned to the same operational configuration present before the failure.

Comment 1: Software maintenance means a standard offering which includes acknowledgement of new and existing problems within 4 hours and a commitment to fix defects within a reasonabletime.

Comment 2: The intent of hardware maintenance pricing is not met by pricing based on the cost to fix specific failures, even if the failure rate is calculated from Mean Time Between Failures (MTBF). The maintenance pricing must be independent of actual failure rate over the five (5) year period, no matter how many failures occur during that period. The intent is to preclude the use of MTBF to directly compute the maintenance cost for this benchmark.

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 whichthe 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.

9.3.4   It is acceptable to incorporate, for pricing purposes, the use of customer spareable and replaceable hardware items under the following conditions:

   a) The conditions of Clause 9.1.7 must be met.
   b) For spares to met the maintenance requirements of a site, an additional 10% of the designated items, with a minimum of two (2), must be priced.
   c) The sponsor must price a support service which provides replenishment onsite within seven (7) days throughout the five-year maintenance period.

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

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

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

10.1.1.3Settings 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.

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

10.1.2   Clause 2 Related Items

10.1.2.1Results of the ACIDity tests (specified in Clause 2) must describe how the requirements were met. If a database different from that which is measured is used for durability tests, the sponsor must include a statement that durability works on thefully loaded and fully scaled database.

10.1.3   Clause 3 Related Items

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

Provide two functional diagrams which show CPUs, storage devices, communication lines, terminals, and the interconnections between these components. The first diagram must correspond to the benchmark configuration and the second diagram must correspond tothe 90-day priced configuration. A separate pair of diagrams must be provided for each reported result. (The diagrams used for clause 10.1.1.4 may already contain this information. In this case, the additional data required below may optionally be shown in tabular form with references to these diagrams.)

As part of each diagram, show the percentage of the total physical database which resides on each storage device for each of the ABTH files and logs. For the benchmark configuration, show database allocation during 8-hour steady state. For the 90-day priced configuration, show database allocation including storage of 90 days of history records. Data which are duplicated (e.g., mirrored) on more than one device must be clearly labeled to show what is duplicated and on which devices.

Two examples are shown below.

Example 1:

Distribution of ABTH files and Logs in Benchmark Configuration

Etherne

┌─────────────┐   ┌─────────────┐
│ Terminal   │   │   │
│ Concentrato │   │   CPU   │
│   │   │   │
└─────────────┘   └─────────────┘

terminals

Disk name: %ABT1

For each disk %ABT1 to %ABT5
Account   20%
Teller   20%
Branch   20%

Disk name: %ABT5

Disk name: %H1
History 100% (during steady state

Disk name: %LOG

physical log: 100%
logical log: 100% (mirrored with logical log on %LOG2)

Disk name: %LOG2

logical log: 100% (mirrored with logical log on %LOG
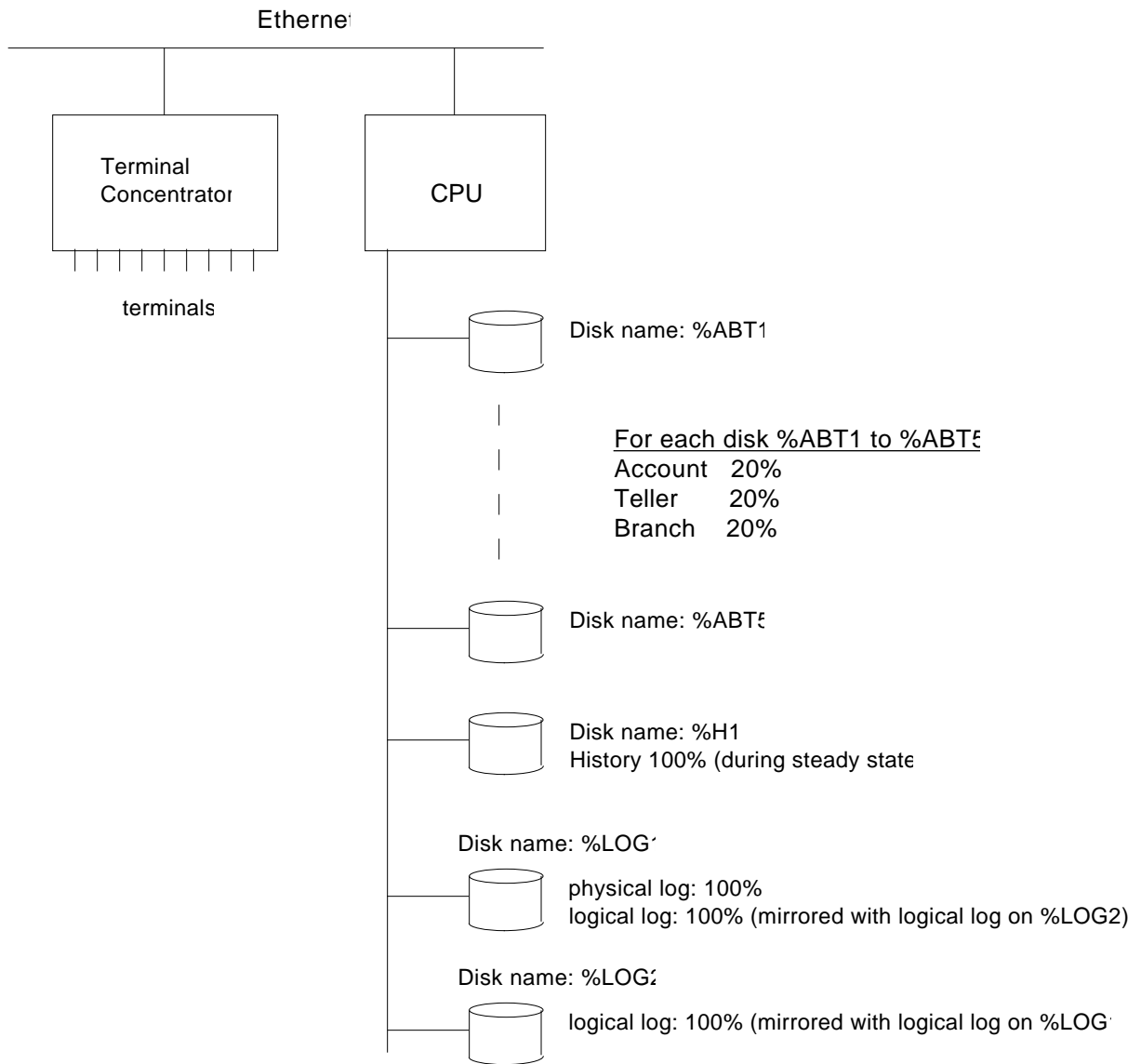
<u>Example 1 (cont.):</u>

<u>Distribution of ABTH files and Logs in Priced Configuration</u>

Ethernet

Terminal
Concentrator

CPU

terminals

Disk name: %ABT1

<u>For each disk %ABT1 to %ABT5</u>
Account   20%
Teller   20%
Branch   20%
History   10%

Disk name: %ABT5

Disk name: %H1

<u>For each disk %H1 to %H5:</u>
History   10%

Disk name: %H5

Disk name: %LOG1

physical log: 100%
logical log: 100% (mirrored with logical log on %LOG2)

Disk name: %LOG2

logical log: 100% (mirrored with logical log on %LOG1
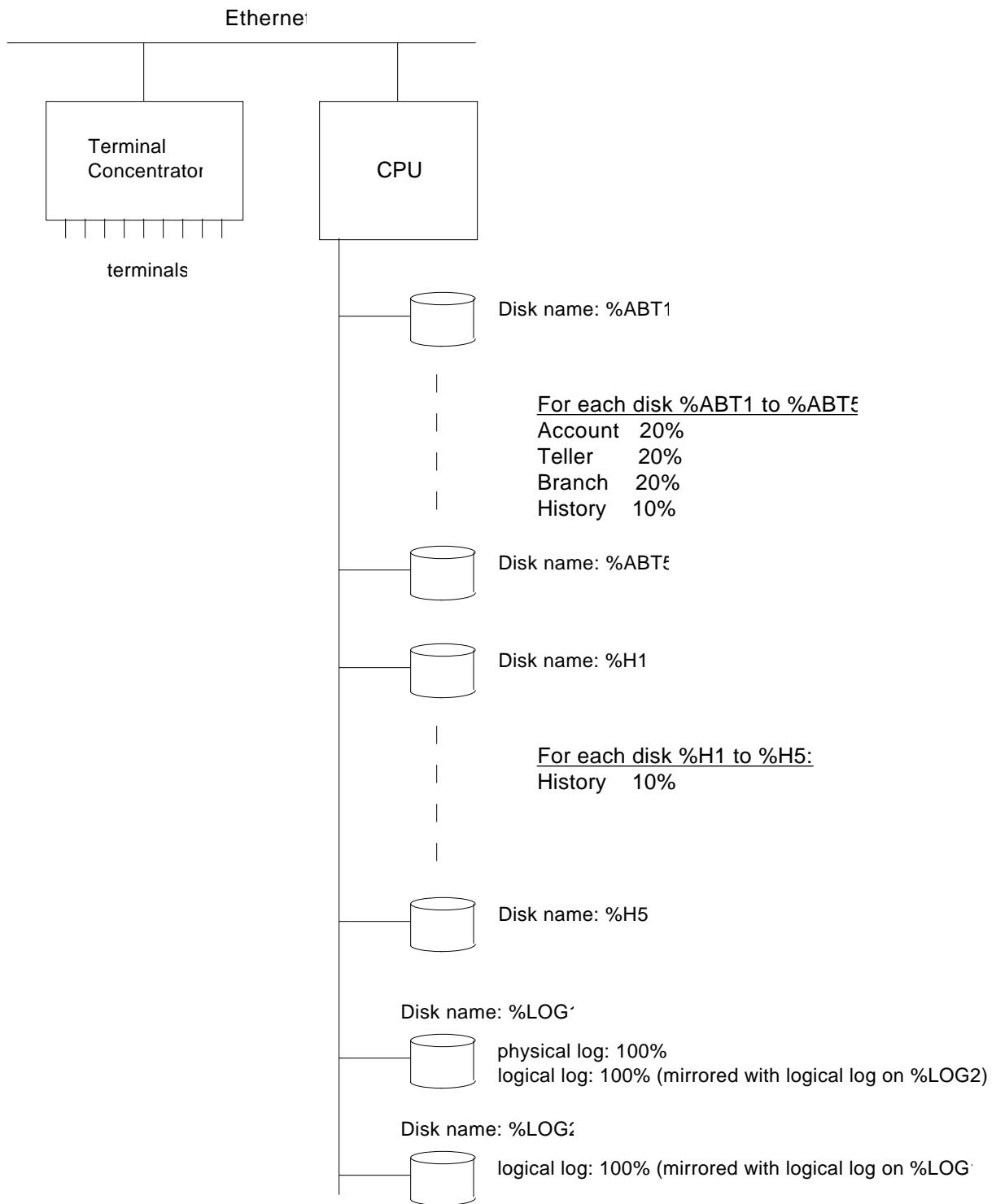
<u>Example 2:</u>

<u>Data Distribution in Benchmarked Configuration</u>

MegaBus

terminals CPU 1

terminals CPU 2

Disk name: /data1

Account: 50%

Disk name: /data3

Account: 25%
Branch: 25%
Teller: 25%
History*: 25%

Disk name: /data2

Branch: 50%
Teller: 50%
History*: 50%

Disk name: /data4

Account: 25%
Branch: 25%
Teller: 25%
History*: 25%

Disk name: /log1 (mirrored disk pair)

audit log: 50%

Disk name: /log2 (mirrored disk pair)

audit log: 50%
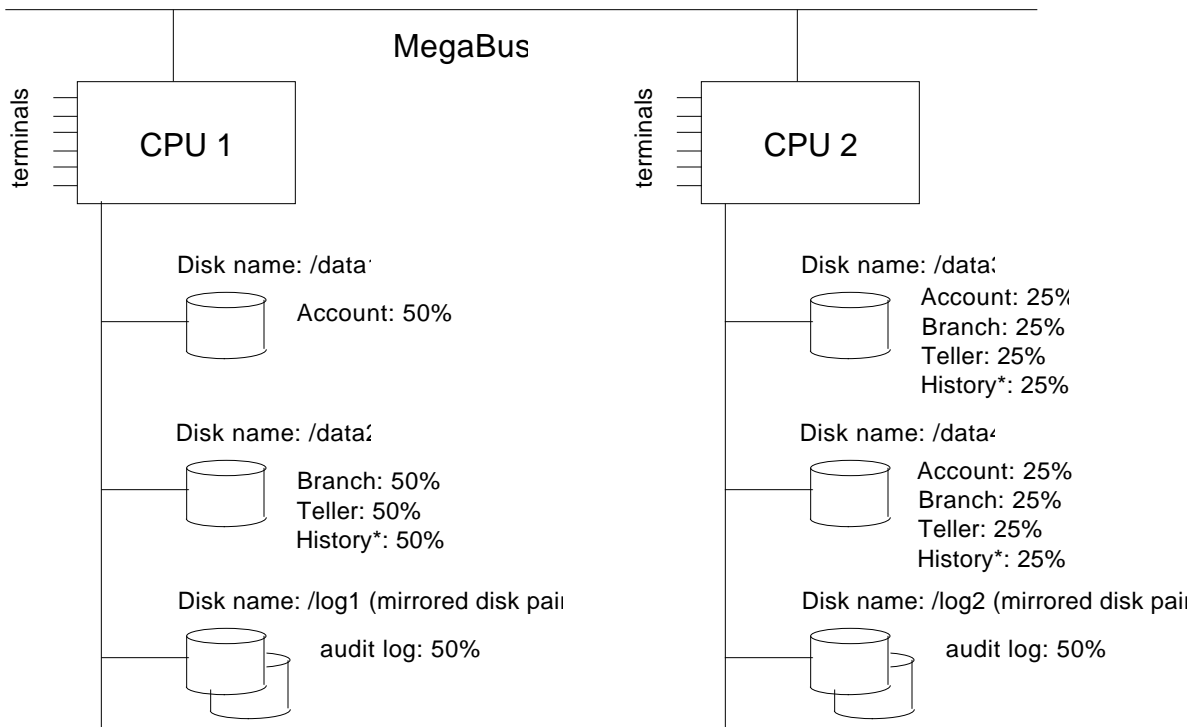
* Distribution of History records represents 8-hour steady state

<u>Data Distribution in Priced Configuration</u>

MegaBus

terminals CPU 1

terminals CPU 2

Disk name: /data1

Account: 50%
History*: 22%

Disk name: /data3

Account: 25%
Branch: 25%
Teller: 25%
History*: 25%

Disk name: /data2

Branch: 50%
Teller: 50%
History*: 28%

Disk name: /data4

Account: 25%
Branch: 25%
Teller: 25%
History*: 25%

Disk name: /log1 (mirrored disk pair)

audit log: 50%

Disk name: /log2 (mirrored disk pair)

audit log: 50%
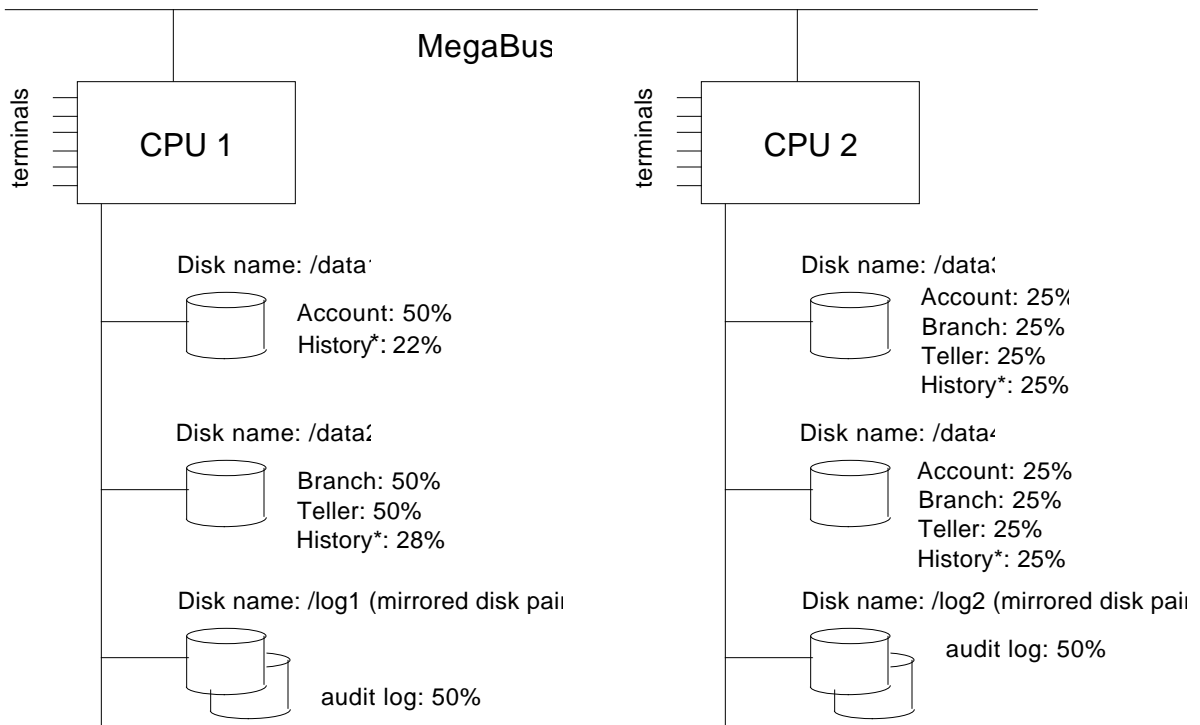
* Distribution of History records represents 90-day storage requirement

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

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

10.1.4   Clause 5 Related Items

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

10.1.4.2Vendors must clearly disclose if horizontal partitioning is used. Specifically, vendors must:
1. Describe textually the extent of transparency of the implementation.
2. Describe which tables/files were accessed using partitioning.
3. Describe how partitioned tables/files were accessed.

The intent of this clause is that details of non-transparent partitioning be disclosed in a manner understandable to non-programmer individuals (through use of flow charts, pseudo code, etc.).

10.1.4.3The sponsor must disclose percentage of remote and home transactions, percentage of remote and foreign transactions, if applicable, and the actual distribution of accounts across the nodes, if applicable.

10.1.5   Clause 6 Related Items

Report all the data specified in Clause 6.6, including reported tpsA, maximum and average response time, as well as performance curves for tpsA vs. response time and response time distribution (see Clauses 6.6.1 and 6.6.2).

10.1.6   Clause 7 Related Items

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

10.1.6.2A 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.

10.1.6.3A description of the method used to determine the reproducibility of the measurement results.

10.1.6.4A statement of the duration of the measurement period for the reported tpsA (it should be at least 15 minutes and no longer than 1 hour).

10.1.7   Clause 8 Related Items

10.1.7.1Disclose the following information related to the RTE:
1. The name of the RTE and whether it is commercially available or proprietary.
2. The hardware on which the RTE runs.
3. The component(s) emulated by the RTE.
4. Commands to start the RTE including pertinent parameters.
5. The type of communication protocol used or simulated between the RTE and SUT.
6. The timing delays associated with the simulation of the components and the communication protocol used.
7. Generation of the success file (used for testing durability).
8. The number of processes per simulated terminal (one process for each terminal or one process per multiple terminals).
9. Generation of random numbers to show that no two simulated terminals will use the same pseudo-random sequence.
10. Listing of input scripts and parameter file to the RTE.
11. Algorithm used to generate transaction input and a sample of that input.
12. Algorithm used to determine delay times between transactions.

13. Benchmark sequencing including ramp-up period, steady state measurement window(s), and transaction success/failure determination and recording.

14. A list and brief description of the data that are collected and the reduction process of that data to determine the results.

10.1.7.2A proof that the functionality and performance of the components being emulated in the Driver System are equivalent to that of the priced system. The sponsor must list all hardware and software functionality of the driver and its interface to the SUT.

10.1.7.3If the SUT contains a WAN or a LAN network, its bandwidth should be specified. The sponsor must describe the network configuration per clause 8.6.5.

10.1.7.4The sponsor must disclose the mean and maximum think times and a graph of the distribution of think times.

10.1.8   Clause 9 Related Items

10.1.8.1A detailed list of hardware and software used in the priced system must be disclosed. Pricing source(s) and effective date(s) of price(s) must also 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.

10.1.8.2The 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.

10.1.8.3The delivery date for general availability (availability date) of products used in the price calculations must be reported. When the priced system includes products with different availability dates, the reported availability date for the priced system must be the date at which all components are committed to be available.

10.1.8.4A statement of the measured tpsA and the calculated price/tpsA.

10.1.8.5Additional Clause 9 related items may be included in the full disclosure report for each country specific priced configuration. Country specific pricing is subject to Clause 9.1.6.

10.1.8.6The basis for the calculation to determine the additional storage space required in Clause 9.2.3.1 must be included.

10.1.8.7For any usage pricing, the sponsor must disclose:
- The usage level at which the component was priced.
- A statement of the company policy allowing such pricing.

Comment: Usage pricing may include, but is not limited to, the operating system and database management software.

10.1.9   Clause 11 Related Items

10.1.9.1~~If the benchmark has been independently audited, then the~~ The auditor's name, address, phone number, and a ~~brief audit summary report~~ lcopy of the auditor's attestation letter 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 whom to contact in order to obtain a copy.~~ Full Disclosure Report.

10.1.9.2 A review of the pricing model is required to ensure that all components required are priced (see Clause 11.3.9). The auditor is not required to review the final Full Disclosure Report or the final pricing prior to issuing the attestation letter.

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 decreases can be reflected in a new published price/throughput. When cumulative price changes have resulted in an increase of 5% or more from the disclosed price/performance, the test sponsor must submit revised price/performance results to the TPC within 30 days of the effective date of the price changes to remain compliant. The benchmark need not be rerun to remain compliant.

Comment: The intent is that the published price/performance reflect actual current price/performance.

10.3.2 Hardware or software product substitutions within the SUT, with the exception of equipment emulated as allowed in Clause 8, require the benchmark to be re-run with the new components in order to re-establish compliance. A new set of testdata mustbe provided, as described in Clause 8.6.4.4, for any substituted equipment that was emulated during the benchmark.

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

10.3.4 A report may be revised to add or delete Clause 9 related items for country specific priced configurations.

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 tpsA rating of the system to change by more than 5%, then the test sponsor will be required to re-validate the benchmark results.

10.3.5 Repricing of current results must be reviewed and approved by the auditor if there is a change to the pricing model. Changes in prices of line item components do not constitute a pricing model change.

10.4 Official Language

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

11.1   An independent audit of the benchmark result~~s is highly recommended. An audit checklist is provided as part of this specification.~~ by an auditor certified by the TPC is required. Please obtain the current audit checklist from one of the auditors.   The term "certified" is defined as: "the TPC has reviewed the qualifications of the auditor and certified that the auditor is capable of verifying compliance of the benchmark result." Please see the TPC Audit Policy for a detailed description of the auditor certification process.

~~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   <u>Clause 1 Related Items</u>

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

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

11.3.2   <u>Clause 2 Related Items</u>

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

11.3.3   <u>Clause 3 Related Items</u>

11.3.3.1For 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.

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

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

11.3.4   <u>Clause 4 Related Items</u>

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

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

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

11.3.5   <u>Clause 5 Related Items</u>

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

11.3.5.2If horizontal partitioning is used, establish whether or not it is transparent to the application program as defined in Clause 10.1.4.2.

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

11.3.6   <u>Clause 6 Related Items</u>

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

11.3.6.2If 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   <u>Clause 7 Related Items</u>

11.3.7.1 Verify 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 tpsA (at least 15 minutes and no longer than 1 hour).

11.3.7.5 Verify that the response time and the tpsA rate 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.~~

11.3.9.1 Verify that the pricing model includes all hardware and software licenses, warranty coverage and additional maintenance costs as required in Clause 9.

Comment 1: The pricing model is a spreadsheet detailing how the 5 year cost of ownership is computed (see Clauses 9.1.5, 10.1.8.1, and 10.1.8.2). It should contain the prices, discounts, warranty information, and maintenance costs for all the hardware and software components in the priced configuration. Letters with price quotes for components from OEM sources must also be verified.

Comment 2: Since final pricing for new products is typically set very close to the product announcement date, the auditor is not required to verify the final pricing of the tested system prior to issuing the attestation letter.

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

11.3.9.2 Verify that the priced configuration includes sufficient storage for the database, history, and recovery logs as specified in Clause 9.2.3, and can be configured in the priced system.

11.3.9.3 Verify that the priced configuration includes sufficient storage for the database, history, and recovery logs as specified in 9.2.3, 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.~~

11.4   The term "audit" may not be used in a full disclosure report unless the auditing agency is independent of the benchmark sponsors. The term "independent" is defined as: 'the outcome of the benchmark carries no financial benefit to the auditing agency other than fees earned directly related to the audit.' The auditing agency cannot have supplied any performance consulting for the benchmark under audit. In addition, the following conditions must be met:
- The auditing agency cannot be financially related to the sponsor. For example., the auditing agency is a dependent division; the majority of its stock is owned by the sponsor.
- The auditing agency cannot be financially related to any one of the suppliers of the measured/priced components, e.g., the DBMS supplier, the terminal or terminal concentrator supplier, etc.

11.5   In the case of audited benchmark results that are used as the basis for a subsequent test sponsor's reported results, the subsequent test sponsor can claim the results are audited if and only if:

1. The auditor ensured the hardware and software products are the same in both configurations.
2. The auditor reviews the test results reported by the subsequent test sponsor and ensures those results match what was reported by the original test sponsor.
3. The auditor can attest to Clause 11.3.9, with the exception of Clauses 11.3.9.1 and 11.3.9.3

```
/*
 *      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 has 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();
void 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(10),
            filler              CHAR(88) DEFAULT SYSTEM
                                );          /* pad to 100 bytes */
exec sql CREATE TABLE tellers (
            Tid                 NUMERIC(9), PRIMARY KEY(Tid),
            Bid                 NUMERIC(9)  FOREIGN KEY REFERENCES branches,
            Tbalance            NUMERIC(10),
            filler              CHAR(84) DEFAULT SYSTEM
                                );           /* pad to 100 bytes */
exec sql CREATE TABLE accounts (
            Aid                 NUMERIC(9), PRIMARY KEY(Aid),
            Bid                 NUMERIC(9) FOREIGN KEY REFERENCES branches,
            Abalance            NUMERIC(10),
            filler              CHAR(84) DEFAULT SYSTEM
                                );           /* pad to 100 bytes */
exec sql CREATE TABLE history (
            Tid                 NUMERIC(9) FOREIGN KEY REFERENCES tellers,
            Bid                 NUMERIC(9) FOREIGN KEY REFERENCES branches,
            Aid                 NUMERIC(9) FOREIGN KEY REFERENCES accounts,
            delta               NUMERIC(10),
            time                TIMESTAMP,
            filler              CHAR(22) 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                                     */
```