TPC BENCHMARK™ B

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
23 August 1990	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.

TABLE OF CONTENTS

0.1 Introduction

TPC Benchmark™ B exercises the database components necessary to perform tasks associated with that class of transaction processing environments emphasizing update-intensive database services. Such environments are characterized by:

- Significant disk input/output
- Moderate system and application execution time
- Transaction integrity

This benchmark is not OLTP in that it does not require any terminals, networking, or think time. This benchmark uses terminology and metrics which are similar to other benchmarks, originated by the TPC and others. The only benchmark results comparable to TPC Benchmark™ B are other TPC Benchmark™ B results. In spite of similarities to TPC-A, TPC-B contains substantial differences which make TPC-B results not comparable to TPC-A.

The metrics used in TPC Benchmark™ B are throughput as measured in transactions per second (TPS), subject to a residence time constraint; and the associated price-per-TPS. The metric for this benchmark is "tpsB". All references to tpsB results must include both the tpsB rate and the price-per-tpsB to be compliant with the TPC Benchmark™ B standard. Comparison of price/performance results disclosed in one country may not be meaningful in another country because of pricing and product differences.

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

The extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC Benchmark™ B approximates the customer application. Relative performance of systems derived from TPC Benchmark™ B 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 systems design and implementation. Relative system performance will vary as a result of these and other factors. Therefore TPC Benchmark™ B should not beused 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.

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

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

1.2 The Transaction Profile

Given Aid, Tid, Bid, Delta by driver (see Clause 1.3 and 8):

```
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
Return Account_Balance to driver
```

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

1.3 Transaction Inputs and Outputs

1.3.1 For each transaction, the driver shall present to the SUT (see Clause 8) at least four distinct fields including Account_ID, Teller_ID, Branch_ID, and Delta. Branch_ID in the input is the identifier of the branch where the teller is located.

1.3.2 Each transaction shall return to the driver the 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 driver, i.e., that the application retrieve the account balance.

1.3.3 The generation of input 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 driver.

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™ B 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™ B 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 the transaction to take the database from one consistent state to another.

2.3.2 Consistency Conditions

A consistent state for the TPC Benchmark™ B 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™ B 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 tpsB (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 ofarbitrary transactions, not just TPC Benchmark™ B 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™ B 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 driver subsequently receiving notification of that fact, since a failure may occur after the transaction commits but before the driver records the transaction.

Comment 2: Although the order of operations in the transaction profile (see Clause 1.2) is immaterial, the return of the Account_Balance to the driver 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 durablemedia. 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 which have returned to the driver have in fact been committed in spite of any single failure from the list in Clause 2.5.3.

parIt 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 the multiple hardware components, whichever is greater. The database subset must be scaled to at least 10% (minimumof 2 tps) of thefully scaled database size. The test sponsor must state that to the best of their knowledge, a fully loaded and fully scaled 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™ B 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.

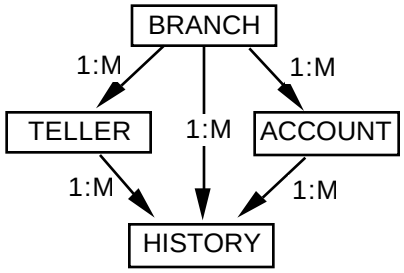
Step 5: Compare the contents of the "success" file and the History file to verify that every record in the "success" file has a corresponding record in the History file. Also verify that the number of records in the History file is greater or equal to the original count, as obtained in Step 1, plus the number of records in the "success" file. If there is an inequality, the History file must contain additional records. (Comment: This difference should be due only to transactions which were committed on the system under test, but which were not recorded by 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.

Comment: If the driver and "success" file are on the SUT, the sponsor must document the method used to make the "success" file an accurate representation of the completed transactions.

3.1 Entities, Relationships, and Characteristics

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



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

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 handled by the database servers, file systems, etc. be of the same size. Therefore, therecords / 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 tpsB 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 numbers since they are simply offsets from the beginning of a file. This does not preclude hashing schemes or other file organizations which have provisions for adding, deleting, and modifying records in the ordinary course of processing. This clause places no restrictions on the History file.

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

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

4.1 The intent of the scaling rules is to maintain a fixed relationshipbetween 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 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)

4.2.1 The ratio of the values in Clause 4.2 must be maintained. Should any value be exceeded, the others must be increased proportionately.

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 tpsB, storage must be maintained for the number of history records specified in Clause 9.2.4.1. 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 9.2.4.1.

4.4 Reported tpsB may not exceed the configured (nominal) rate represented by the file / table sizes in Clause 4.2. While the reported tpsB may fall short of the maximum allowed by the configured system, the price per tpsB computation must report the price of 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 of 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 of the transaction's teller.

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, details of 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™ B data operation accesses approximately 100 bytes, not some smaller, proper subset.

5.3 Transaction Input Generation

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

5.3.2 If multiple driver processes (see Clause 8) are implemented, then each driver process must generate transaction input.

5.3.3 Teller_ID Generation

For each transaction the driver (or drivers) must choose a teller at random from some range of the Teller_ID values without regard for any concurrently pending transactions. For single node systems, the teller for each transaction must be randomly chosenfrom the entire range of tellers. In the case of a multinode system in which the database is partitioned among nodes of a cluster or network, the tellermay be randomly chosen from a subset of the total range of tellers corresponding to all the tellers on a single node, but no fewer.

5.3.4 Branch_ID Generation

Given the randomly chosen teller from Clause 5.3.3, the corresponding branch is determined (i.e., each teller resides at a known branch).

5.3.5 Account_ID Generation

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 divided evenly between two nodes, approximately 7.5% (1500 / 2999 * 15%) of the transactions cause cross-node activities. With the same 3000 branches divided among three nodes, approximately 10% (2000 / 2999 * 15%) cause cross-node activities, etc. Note that 2999 is used since the home branch by definition does not qualify.

5.3.6 Delta Amount

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

5.3.7 Random Distribution

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" is the steady state period (see Clause 7.1) during the execution of the benchmark for which the test sponsor is reporting a tpsB number and residence time data. The term "completed transaction" is a transaction which has been successfully committed at the SUT and whose output has been recorded at the driver (see Clause 8.4).

6.1.2 Each transaction must be individually timed.

6.2 Residence Time Definition

Residence time of a transaction is defined by:

RT = T2 - T1

where T1 and T2 are defined as:

T1 = time stamp taken by the driver before supplying transactional inputs to the SUT

T2 = time stamp taken by the driver after receiving corresponding response

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

Comment: Since the driver program does not represent a user terminal, residence times achieved in this benchmark do not represent response times at the stated throughput levels in an OLTP environment.

6.3 Residence Time Constraint

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

Comment: This residence time criterion has been chosen to provide a single criterion for all configurations, and in particular for very-low throughput systems.

6.4 Computation of tpsB Rating

6.4.1 The reported tpsB is the total number of committed transactions which both started and completed during the measurement interval, divided by the elapsed time of the interval. The reported tpsB can be no greater than the measured tpsB.

6.4.2 For reporting the throughput of the SUT in units of transactions per second, the terminology should be "tpsB". This metric is NOT comparable to other TPS metrics except "tpsB".ar6.4.3 Reported tpsB numbers must be expressed to exactly two decimal places, rounded to the hundredth place.

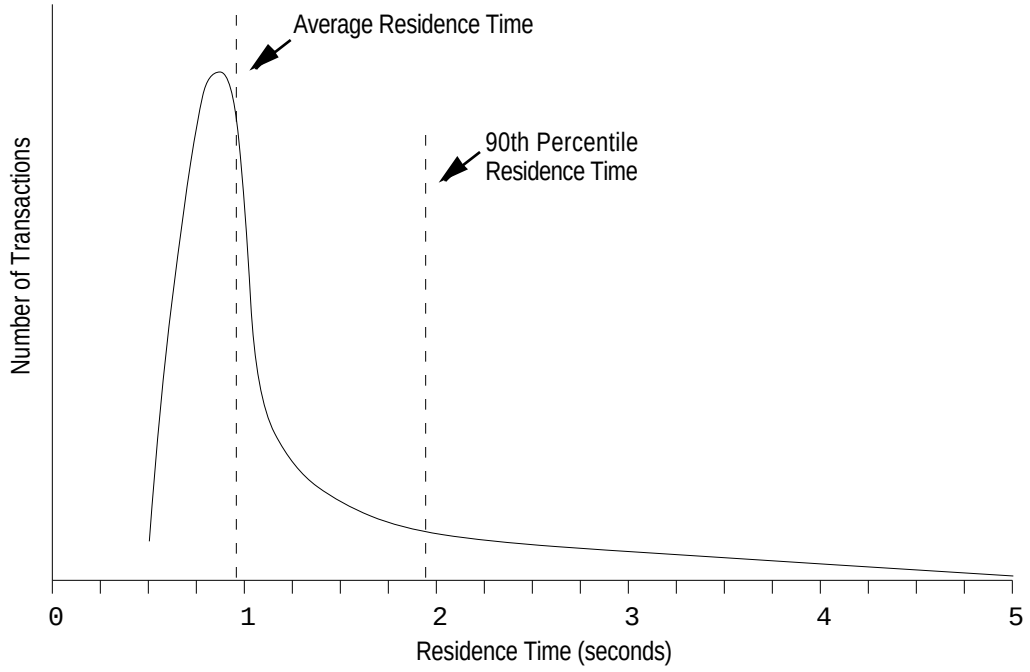
6.5 Interpolation and Extrapolation Prohibited

The reported tpsB rate must be measured rather than interpolated or extrapolated. For example, suppose 9.13 tpsB is measured during which 90% of the transactions completed in less than 1.7 seconds and 9.77 tpsB is measured during which 90% of the transactions completed in less than 2.3 seconds. Then the reported tpsB 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 residence times of transactions started and completed during the measurement interval must be reported.The range of the X axis must be from 0 to 5 seconds residence time. At least 20 equal non-overlapping intervals must be reported. A sample is shown in Graph 1. The maximum and average residence times must also be reported.

Graph 1: Residence Time Distribution



6.6.2 The percentage of **home** transactions must be between 84% and 86% and the percentage of **remote** transactions must be between 14% and 16% of the total completed transactions in the measurement interval. (See Clauses 5.1 and 5.3.5) Report the percentage of transactions that started and completed in the measurement interval that are home and remote transactions.

6.6.3 The percentage of transactions that started but did not complete during the measurement interval must be less than 1% of the total that started. This number must be reported.

6.6.4 For systems that process remote transactions on two nodes (see Clause 5.1.3), report the percentage of **remote** transactions that started and completed during the measurement interval and involved a **foreign** node. Also, report the frequency distribution, maximum, and average residence times of these transactions.

6.6.5 TPS Stability Test

It must be demonstrated that the configuration, when tested at the reported tpsB rate, has the property of stable throughput despite small changes in the number of concurrently active transactions. This is to detect atypical throughput characteristics ofthe reported configuration.

Comment: In well-behaved systems, increases in the number of concurrently active transactions at the throughput limit result in increases in residence time while throughput (i.e., TPS) remains approximately constant.

For the reported tpsB rate T and residence time R, the average number of concurrently active transactions, C, can be calculated as:

$$C = T \cdot R$$

where:

C = average number of concurrently active transactions

T = reported tpsB rate

R = average residency time for transactions in the measured interval

If T_R is the reported tpsB rate and R_R is the reported average residence time, then the corresponding C_R must be reported. Also, two additional values of C must be reported and shown graphically as in the example below.

These are C_L and C_H, and must be in the following range:

$$.7\ C_R \leq C_L \leq .8\ C_R$$

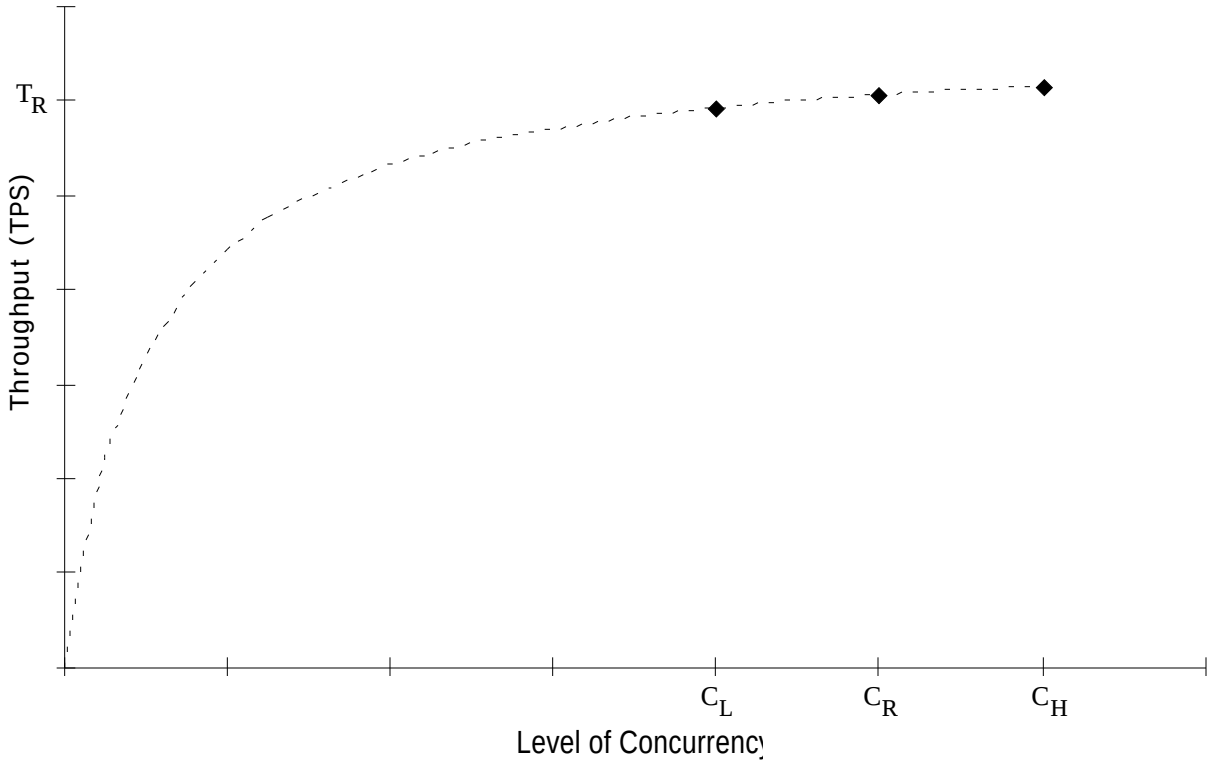
$$C_H \geq 1.2\ C_R$$

Examples of how these two values are obtained include varying the number of transaction generators and changing the number of configured threads in a TP monitor.

Comment: C_H must be measured on the same configuration as C_R, but the corresponding T_H is not required to meet the residence time constraint.

An example of these results is illustrated in Graph 2.

Graph 2: TPS vs. Concurrency



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 tpsB 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 tpsB 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 reproducibilityof 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.

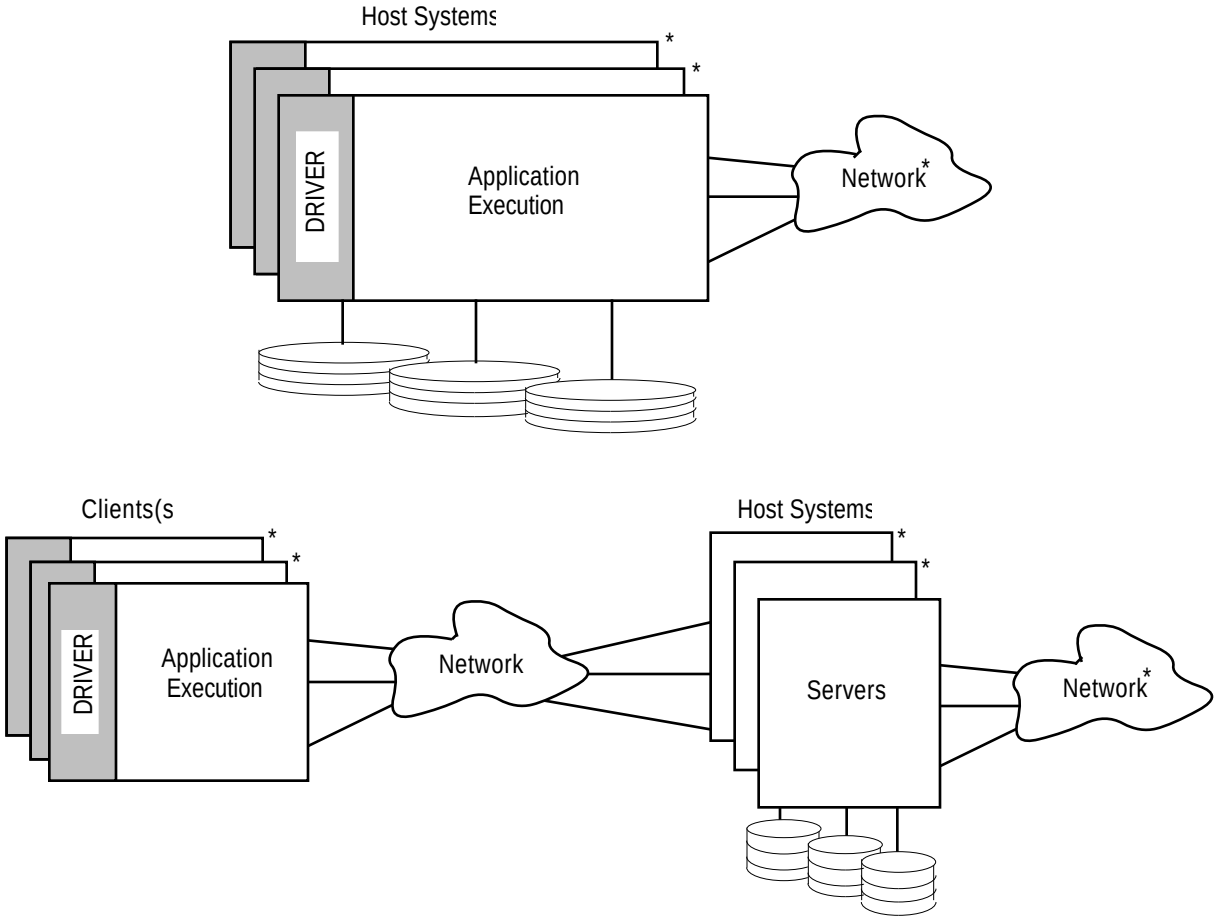
Note to Comment 2: Some systems defer writes of changed pages/blocks to the durable-medium-resident database. Such systems can maintain buffers/caches in a volatile medium (e.g.,memory) for use by the DBMS, operating system, and disk control system, which are not synchronized with the durable-medium-resident 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**

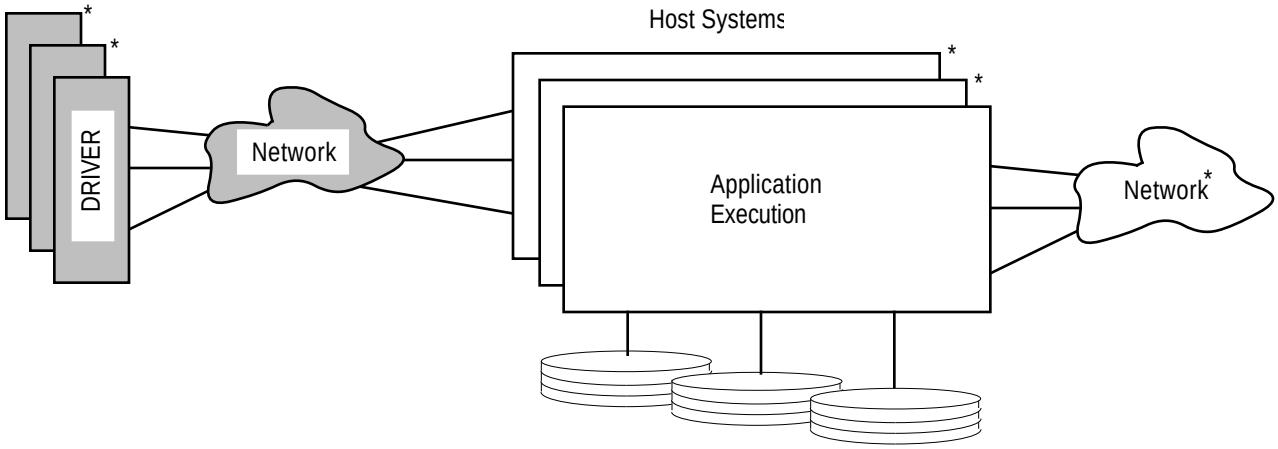
The driver presents transactions to the system under test (SUT). The SUT executes these transactions and replies to the driver. There are two forms of drivers: an **internal driver** resides on the SUT hardware and software. An **external driver** resides on a separate hardware and software complex, and typically communicates with the SUT via a communications network using a client/server remote procedure call mechanism. The following diagrams illustrate these two options. The driver is the shaded area and the SUT is the unshaded area. The diagram also depicts the driver/SUT boundary (see Clause 8.3 and 8.4) where residence time is measured:

Items marked by an "*" are optional.

Two internal driver/SUT systems (a "mainframe" and a client/server configuration):



An external driver/SUT system:



8.2 Test Configuration

The test configuration consists of the following elements:

- System Under Test (SUT)
- Driver System

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 application described in Clause 1, and whose aggregate performance will be described by the metric tpsB.
- Any front-end systems needed to communicate with an external driver system.
- The host system(s) including hardware and software supporting the database employed in the benchmark.
- 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).r8.3.2 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 As stated in Clause 8.1, an internal or external driver presents a transaction load to the SUT.

8.4.2 The driver is a logical entity that can be implemented using one or more physical programs, processes, or systems. Each driver process must perform all of the following functions:

- Supplies the following transactional inputs to the SUT: Account_ID, Teller_ID, Branch_ID, and delta, and no other information;
- Receives transaction responses (new Account_Balance) from the SUT;
- Takes timestamps (T1 and T2) and records either T1 and T2 or RT as defined in Clause 6.2;

Comment: The driver may do statistical accounting of transaction residence times.

8.4.3 The driver, whether internal or external, is expected to do only the functions specified in Clause 8.4.2. Any other function performed by the driver must be: (1) justified as to why it is included (e.g., it is an integral part of a commercial driver product); (2) demonstrated to not improve performance. In addition, the code for

any functionality not specified in Clause 8.4.2 must be disclosed unless it can be demonstrated to bea proprietary part of a commercially available and supported product. The intent is to prevent the use of special purpose driver code to enhance performance.

8.4.4 Any software or hardware used exclusively by the external driver (i.e. not used by the SUT) is not considered as part of the SUT.

Comment: The intent is that external drivers not priced with the SUT are allowed to perform only the functions outlined in Clause 8.4.2. For example, inclusion of procedure names or data locations by the external driver is prohibited.

8.5 Further Requirements on the SUT and Driver System

8.5.1 No Database on External 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 external driver during the test. Synchronization between driver and SUT (e.g., through known initial values for ABT balances) is equally disallowed.

8.6 Any TPC-B results are comparable to other TPC-B results regardless of the form of driver (internal or external).

9.1 **Pricing Methodology**

9.1.1 The intent of this section is to define the methodology to be used in calculating the price/tpsB. 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 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 on-line storage for 30 days of history records;

c) Price of additional products that are required for the operation, administration or maintenance of the priced system;

d) 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. This is allowed, but it specifically excludes any use of "one of a kind" hardware/software configurations which the vendor does not intend to ship in the future. Products must be generally available in the country where the SUT is priced.

9.1.4 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 less than 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, thebasis 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 SUT

The entire price of the SUT as configured during the test must be used, including all hardware (new purchase price), software (license charges) and hardware/software maintenance charges over a period of 5 years (60 months).

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: Any usage pricing for TPC-B should be that the number of users is at least the level of concurrency (C_R), rounded up to the next integer. (See Clause 6.6.5.)

9.2.2 The price of hardware and software used *exclusively* by the driver is not included in the price calculation.

9.2.3 Network Pricing

In a distributed system and in a client/server system, the cost of all communications components within the SUT excluding LAN or WAN direct connect cables must be priced.

Comment: The intent is that all components including PADS (packet assemblers-disassemblers), modems, concentrators, multiplexors, etc., required to attach clients and servers or to attach network nodes should be priced.

9.2.4 History Storage and Recovery Log Pricing

9.2.4.1 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 30.

Comment: On-line storage includes magnetic discs, magnetic tapes, optical discs, and any combination of these. Storage is considered on-line if any record can be accessed randomly within one second.

9.2.4.2 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 9.2.4.1.)

9.2.4.3 If it is necessary to price any additional storage devices to fulfill the thirty (30) day history storage requirement, such devices must be of the type(s) actually used in the SUT during the test, and must satisfy the normal system configuration rules.

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

9.2.4.4 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 Clauses 2.5.3.2 and 2.5.3.3 are on-line.

9.2.5 Additional Operational Components

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

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

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

9.2.6 Additional Software

9.2.6.1 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.

9.2.6.2 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: In client/server designs based on workstations, maintenance and warranty terms for 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™ B 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.

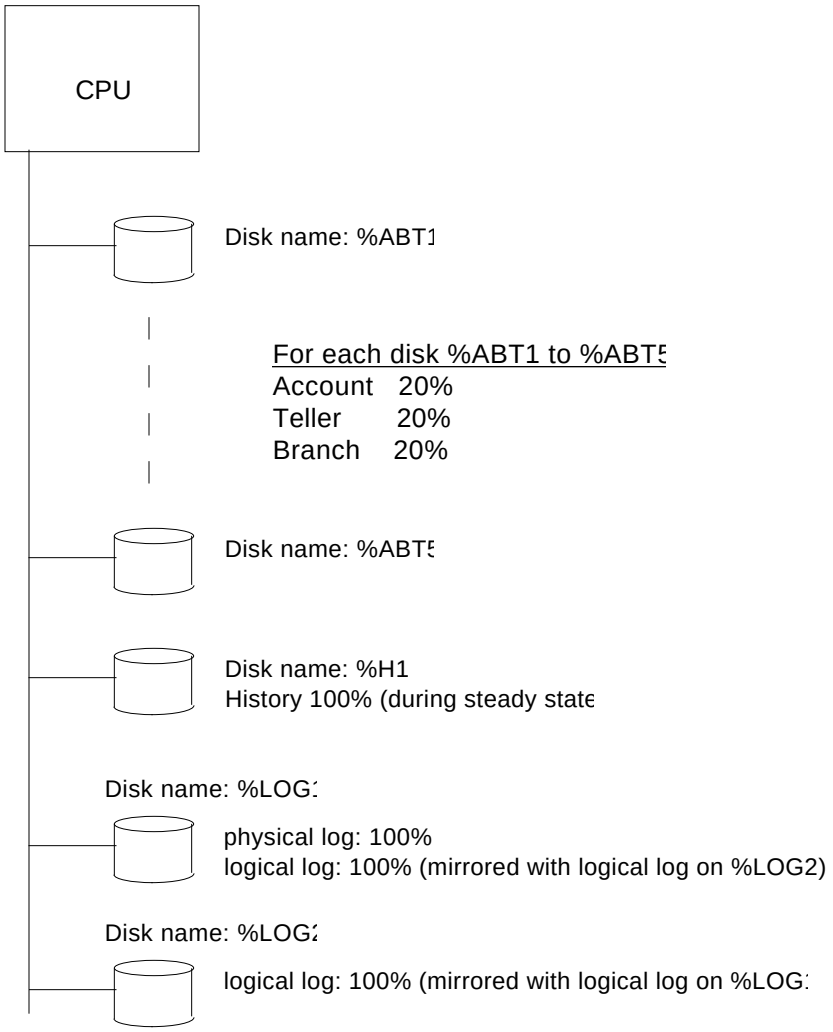
Provide two functional diagrams which show CPUs, storage devices, and the interconnections between these components. The first diagram must correspond to the benchmark configuration and the second diagram must correspond to the 30-day priced configuration. A separate pair of diagrams must be provided 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 30-day priced configuration, show database allocation including storage of 30 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.

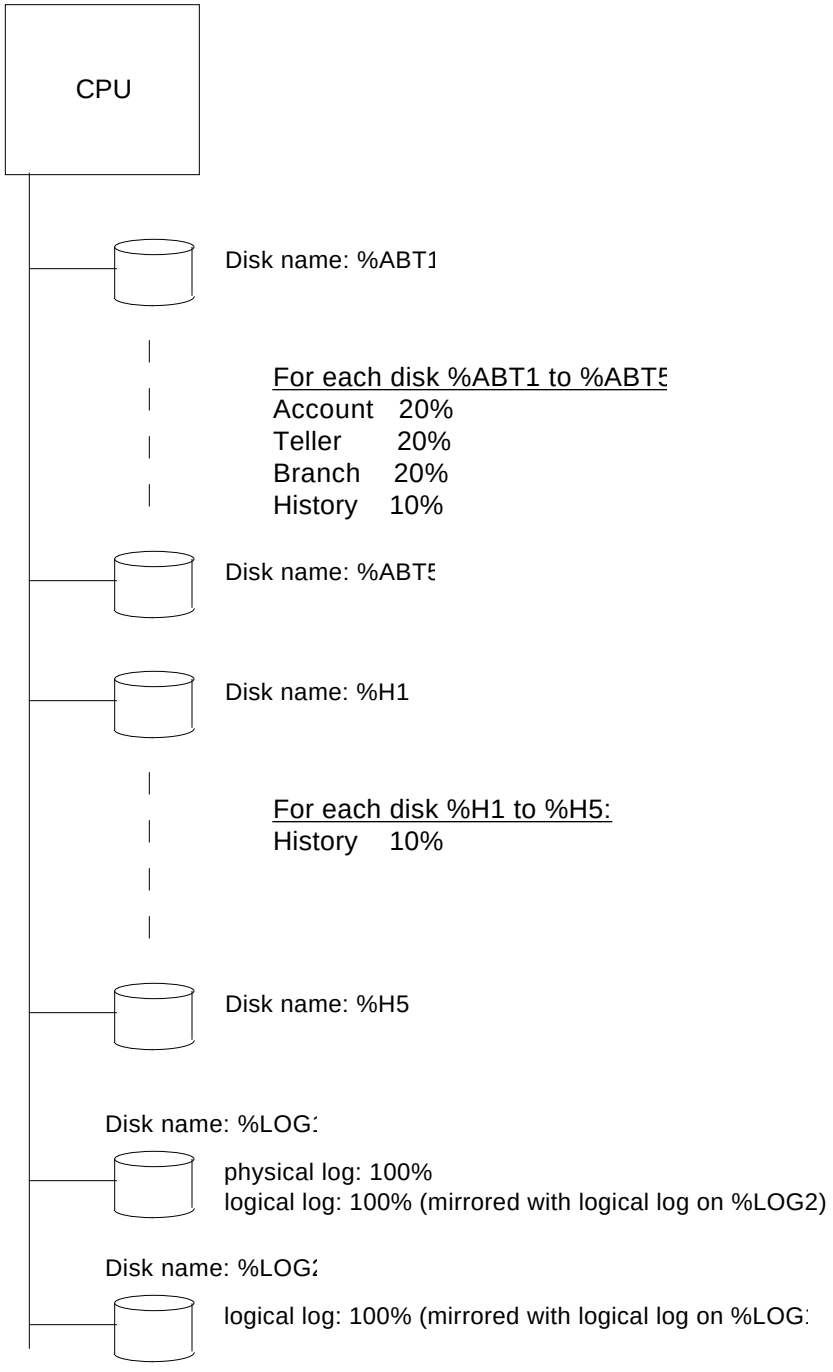
Example 1:

Distribution of ABTH files and Logs in Benchmark Configuration



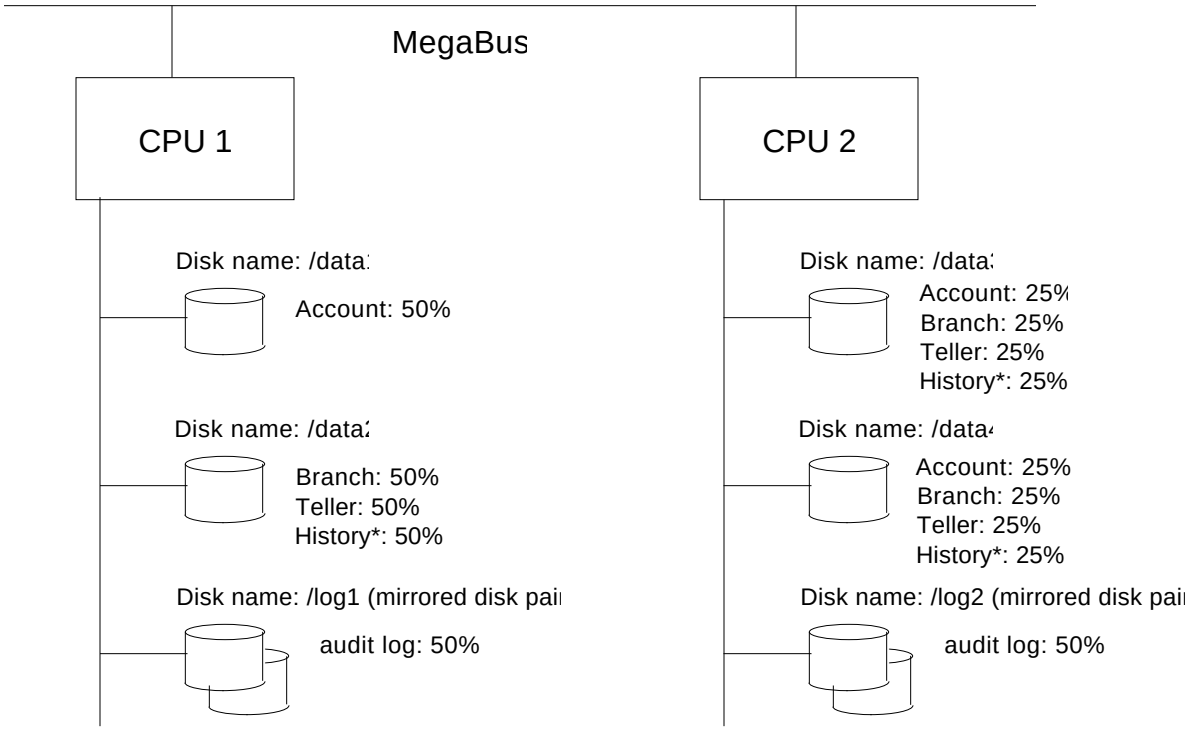
Example 1 (cont.):

Distribution of ABTH files and Logs in Priced Configuration



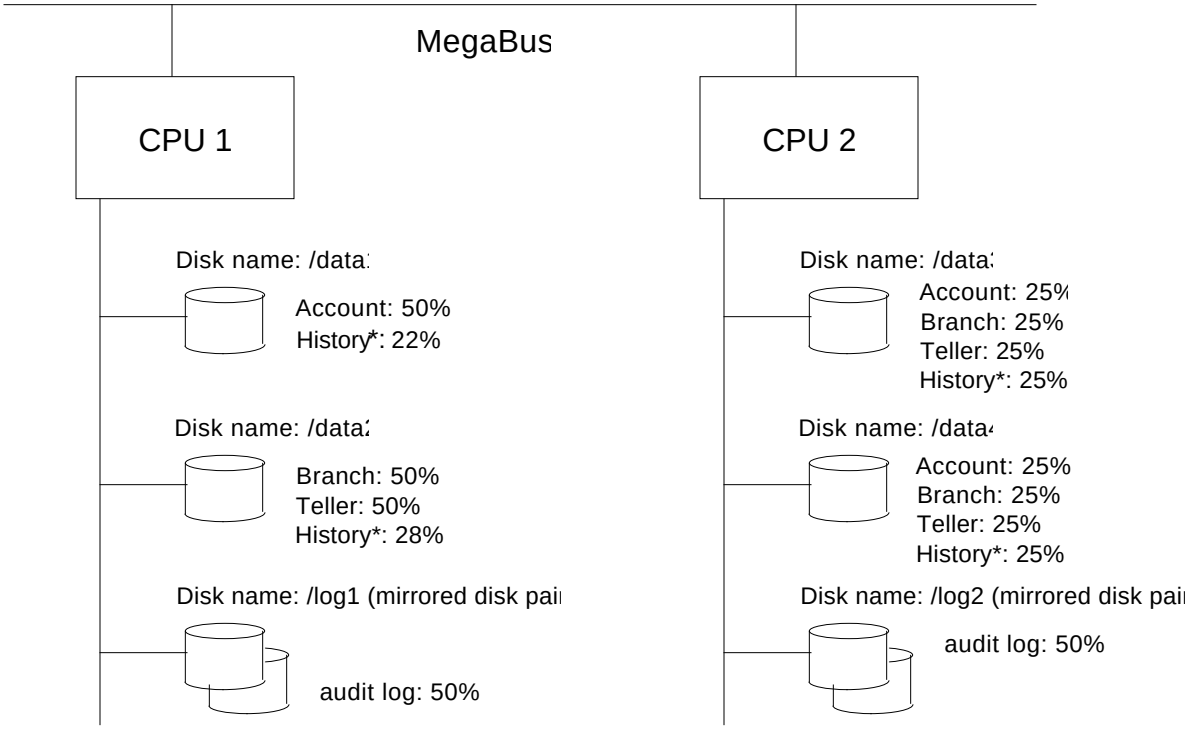
Example 2:

Data Distribution in Benchmarked Configuration



* Distribution of History records represents 8-hour steady stat

Data Distribution in Priced Configuration



* Distribution of History records represents 90-day storage requireme

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.).

10.1.5 Clause 6 Related Items

10.1.5.1Report all the data specified in Clause 6, including measured and reported tpsB, maximum and average residence time, as well as performance curves for number of transactions vs. residence time (see clause 6.6.1) and throughput vs. level of concurrency for three data points (see clause 6.6.5). Also, the sponsor must include the percentage of home and remote transactions, the number and percentage of in-process transactions, and the percentage of remote and foreign transactions, ifapplicable.

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 tpsB (it should be at least 15 minutes and no longer than 1 hour).

10.1.7 Clause 8 Related Items

10.1.7.1If the driver is commercially available, then its inputs should be specified. Otherwise, a description of the driver should be supplied.

10.1.7.2A complete functional diagram of the hardware and software of the benchmark configuration including the driver must be provided. The sponsor must list all hardware and software functionality of the driver and its interface to the SUT.

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 committed 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 tpsB, and the calculated price/tpsB.

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 9.2.4.1 must be included.

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™ B", 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 tpsB 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 results ~~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 Clause 1 Related Items

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

11.3.1.2 Verify that transaction inputs and outputs satisfy Clause 1.3.

11.3.2 Clause 2 Related Items

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

11.3.3 Clause 3 Related Items

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

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

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

11.3.4 Clause 4 Related Items

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

11.3.4.2 Verify 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 tpsB rating per Clause 4.4 (scaling).

11.3.5 Clause 5 Related Items

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

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

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

11.3.6 Clause 6 Related Items

11.3.6.1 Verify the method used by the driver to measure the residence time.

11.3.7 Clause 7 Related Items

11.3.7.1 Verify that the SUT had reached a steady state prior to commencing the measurement interval using the method required by Clause 10.1.6.1.

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.

11.3.7.3Verify the reproducibility of the tpsB rating using the method required by Clause 10.1.6.3.

11.3.7.4Verify the duration of the measurement period for the reported tpsB rate (at least 15 minutes and no longer than 1 hour).

11.3.7.5Verify that the residence time and the tpsB rate were measured in the same time interval.

11.3.8 Clause 8 Related Items

11.3.8.1Verify that the SUT is what is claimed and that the driver performs only driver functions.

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 9.2.4, and can be configured in the priced system.

11.3.9.3Verify that the priced configuration includes sufficient storage for the database, history, and recovery logs as specified in 9.2.4, 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 B coded in ANSI C and ANSI SQL2.
 *     Any equivalent implementation is equally acceptable.
 *
 *     Exceptions:
 *           1.     ANSI/ISO SQL has no explicit BEGIN WORK (begin transaction).
 *                  To show that the driver is outside the transaction,
 *                  explicit BEGIN WORK statements are included
 *           2      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,...)
 *           3.     To clarify the schema, the following SQL2 features are used:
 *                        Primary Key
 *                        Foreign Key
 *                        DateTime datatype
 *                        Default values (to simplify handling of pad chars).
 *           4.     For simplicity, the program does no error checking or handling.
 */

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

/* tpc bm b 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       =  864000;   /* number of history recs in 1 tps db   */

/* working storage                                    */

long i,sqlcode, Bid, Tid, Aid, delta, Abalance;

exec sql END DECLARE SECTION;

void CreateDatabase();
long DoOne(long Bid, long Tid,  long  Aid, long delta);
#include <stdio.h>
/* main program,    creates a 1-tps database:  i.e. 1 branch, 10 tellers,...
 *                    runs one TPC BM B transaction
 */
main()
{
       CreateDatabase();
       Abalance = DoOne(1,1,1,100);     /* add 100 to account 1 by teller 1 at branch 1,
                                           return new balance     */
}
```

```
/*
 *  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 B 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 B transaction.
 */

long DoOne(long Bid, long Tid,  long  Aid, long delta)
{

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;

return ( Abalance);

}                   /* end of DoOne                  */
```