

---

**Hewlett-Packard Company**  
**High Performance Systems Division**

**HP 9000 V2250 Enterprise Server**  
**Using HP-UX 11.00 64-bit and Sybase Adaptive Server Enterprise 11.5**

---

**TPC Benchmark<sup>®</sup> C**  
**Full Disclosure Report**

**First Edition**  
**February 13, 1998**

---

First Edition - February 13, 1998.

Hewlett-Packard Company believes that the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. Hewlett-Packard Company assumes no responsibility for any errors that may appear in this document.

The pricing information in this document is believed to accurately reflect the current prices as of the publication date. However, Hewlett-Packard Company provides no warranty of the pricing information in this document.

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® C should not be used as a substitute for a specific customer application benchmark when critical capacity planning and/or product evaluation decisions are contemplated.

All performance data contained in this report was obtained in a rigorously controlled environment. Results obtained in other operating environments may vary significantly. Hewlett-Packard Company does not warrant or represent that a user can or will achieve similar performance expressed in transactions per minute (tpmC®) or normalized price/performance (\$/tpmC®). No warranty of system performance or price/performance is expressed or implied in this report.

© Copyright Hewlett-Packard Company 1998.

All rights reserved. Permission is hereby granted to reproduce this document in whole or in part provided the copyright notice printed above is set forth in full text on the title page of each item reproduced.

Printed in U.S.A., February 13, 1998.

HP, HP-UX, HP C/HP-UX, HP 9000 are registered trademarks of Hewlett-Packard Company.

Sybase Adaptive Server Enterprise and Sybase Open Client DB-Library are registered trademarks of Sybase, Inc.

TUXEDO is a registered trademark of BEA System , Inc.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

TPC Benchmark, TPC-C, and tpmC are registered certification marks of the Transaction Processing Performance Council.

All other brand or product names mentioned herein are trademarks or registered trademarks of their respective owners.

## *Abstract*

---

### **Overview**

This report documents the methodology and results of the TPC Benchmark<sup>®</sup> C test conducted on the HP 9000 V2250 Enterprise Server in a client/server configuration, using Sybase Adaptive Server Enterprise 11.5<sup>™</sup> and the TUXEDO 4.2.2 transaction monitor. The operating system used for the benchmark was Hewlett-Packard's HP-UX 11.00 64-bit. The application was written in C and compiled using HP C/HP-UX.

### **TPC Benchmark<sup>®</sup> C Metrics**

The standard TPC Benchmark<sup>®</sup> C metrics, tpmC<sup>®</sup> (transactions per minute), price per tpmC<sup>®</sup> (five year capital cost per measured tpmC<sup>®</sup>), and the availability date are reported as required by the benchmark specification.

### **Standard and Executive Summary Statements**

Page *iv* contains the standard system summary and pages *v-vi* contain the executive summary of the benchmark results for the HP 9000 V2250 Enterprise Server.

### **Auditor**

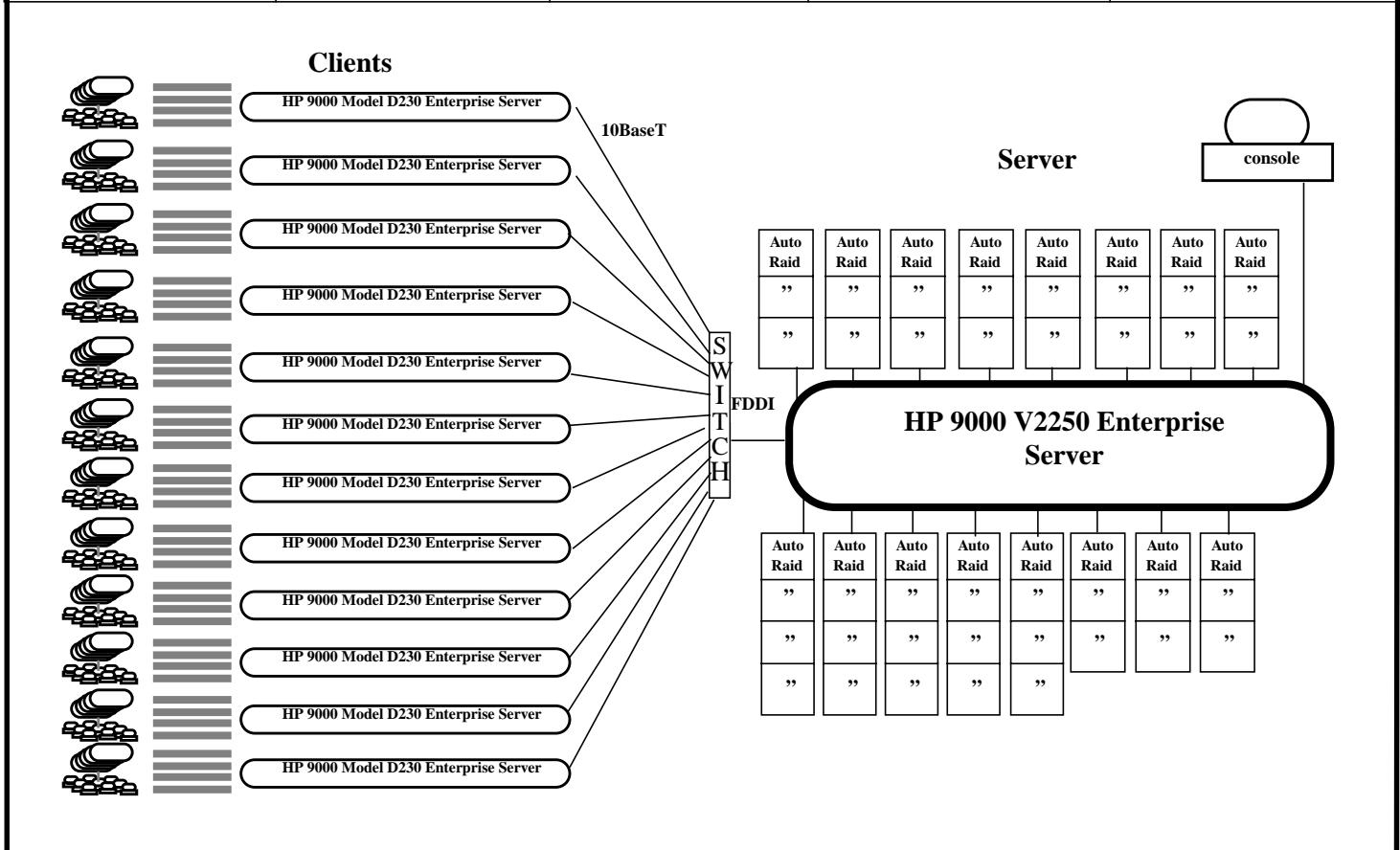
The benchmark configuration, environment and methodology used to produce and validate the test results, and the pricing model used to calculate the cost per tpmC<sup>®</sup>, were audited by Richard Gimarc of Performance Metrics, Inc. to verify compliance with the relevant TPC specifications.

### Standard System Summary

Company Name	System Name	Database Software	Operating System Software
Hewlett-Packard Co.	HP 9000 V2250 Enterprise Server	Sybase Adaptive Server Enterprise 11.5 EBF 7817	HP-UX 11.00 64-bit Extension Pack 9808
HP H/W Availability Date —March 2, 1998 Software Availability Date —August 13, 1998			

Total System Cost	TPC-C <sup>®</sup> Throughput	Price/Performance
Hardware software 5-year maintenance	Sustained maximum throughput of system running TPC-C <sup>®</sup> expressed in transactions per minute	Total system cost/tpmC <sup>®</sup> (\$4,229,894 / 52,117.80)
<b>\$4,229,894</b>	<b>52,117.80 tpmC</b>	<b>\$81.17 per tpmC<sup>®</sup></b>

<b>Hewlett-Packard Co.</b>	<b>HP 9000 V2250 Enterprise Server Client/Server with 12 D230 front-ends</b>		TPC-C® Rev 3.3.2
			Report Date: February 13, 1998
Total System Cost	TPC-C® Throughput	Price/Performance	Availability Date
<b>\$4,229,894</b>	<b>52,117.80 tpmC®</b>	<b>\$81.17 / tpmC®</b>	<b>August 13, 1998</b>
Processors	Database Manager	Operating System	Other Software
16 PA-RISC 8200 240 MHz	Sybase Adaptive Server Enterprise 11.5 EBF 7817	HP-UX 11.00 64-bit Extension Pack 9808	TUXEDO Transaction Monitor 4.2.2
			Number of Users
			43,000



System Components	Server (V2250)		Each Client of 12 (D230)	
	Qty	Type	Qty	Type
Processors	16	240 MHz PA-RISC 8200 2 MB I-cache, 2 MB D-cache	1	160 MHz PA-RISC 7300LC 1 MB combined L2 cache
Memory	16	GB	1	768 MB
Disk Controllers	16	HP-PCI FWD SCSI-2	1	HP-HSC
Disk Drives	53	HP AutoRAID Arrays	1	4 GB Disk
Total Storage (GB)	3,232	GB		
Tape Drives	1	DDS Storage System		
Terminals	1	Console terminal	1	Console terminal

**Hardware and Software Pricing**

Description	Part Number	3d Party		Unit Price	Qty.	Extended Price	5 Year Maint. \$
		Brand	\$				
<b>Server Hardware</b>							
HP 9000 V2250 Enterprise Server	A5081A		1	89,060	1	89,060	100,944
Add'l 240 MHz PARISC 8200 CPUs	A5066A Opt. 0D1		1	24,090	16	385,440	78,566
System Mgmt. Station	A4802A		1	8,059	1	8,059	
Memory carrier	A3730A		1	6,570	4	26,280	
128MB DIMM for a total of 1 GB	Option UTC		1	11,680	16	186,880	
Six slot PCI Card Cage	A3729A, Opt. 0D1		1	4,380	3	13,140	
PCI FWD SCSI-2 Card	A4800A, Opt. 0D1		1	876	16	14,016	
PCI Dual FDDI LAN Adapter	A3739A Opt 0D1		1	2,756	1	2,756	
PCI FDDI LAN Adapter S/W & Lic.	J3626AA		1	292	1	292	
4GB DDS-2 DAT Drive	A3183A, Opt 0DZ		1	1,205	1	1,205	
HP-UX 11.0 Sys Media, CD-ROM	B3920EA, Opt AAF		1	380	1	380	
PCI 64-port Mux card	J3593A Opt. 0D1		1	1,091	1	1,091	
16 port RS232 DB25 port module	J2485A		1	642	1	642	
25 ft cable	J3595A		1	110	1	110	
5.5 kVA UPS	A3589A Opt. 002		1	7,300	1	7,300	2,700
1.3kVA HP UPS Rackmount	A2996A Opt 002		1	1,825	14	25,550	2,419
1.6m Field Integrated Cabinet	C2786A		1	1,318	14	18,452	
Auto Raid Array Model 12H	A3700AZ		1	3,504	53	185,712	254,267
Two 96 MB controllers with Auto Raid	Opt. 203		1	11,972	53	634,516	
Twelve 4.3 GB disk modules	Option 112		1	12,614	27	340,578	
Twelve 9.1 GB disk modules	Option 132		1	19,710	26	512,460	
0.9m 68-pin high density male connec.	Option 801		1	88	37	3,256	
2.5m HDTS68 to HDTS68 Cable	C2924A		1	102	16	1,632	
<b>Subtotal</b>						<b>\$2,458,807</b>	<b>\$438,896</b>
<b>Server Software</b>							
Sybase Adaptive Server	Runtime	Sybase	2	295,000	1	295,000	244,316
Sybase development		Sybase	2	9,600	1	9,600	
Sybase open client		Sybase	2	795	1	795	
Sybase discount		Sybase	2			-30,540	
<b>Subtotal</b>						<b>\$274,855</b>	<b>\$244,316</b>
<b>Client Hardware</b>							
Hewlett Packard 9000 Model D230 Client	A3574A		1	6,169	12	74,028	29,102
1 MB cache	A3586A Opt. 0D1		1	1460	12	17520	
256 MB high-density memory	A3564A Opt 0DZ		1	4234	36	152424	
Console	C1064GX		1	393	12	4,716	1,296
4 GB internal disk	A3643A Opt 0D1		1	1,132	12	13,584	7,517
Combined 10baseT/100VG adapter card	A3402A Opt 0D1		1	583	48	27,984	
<b>Subtotal</b>						<b>\$290,256</b>	<b>\$37,915</b>
<b>Client Software</b>							
HP C/ANSI C Compiler	B3901BA, Option AH0		1	1,150	1	1,150	1,757
BEA Tuxedo 4.2		Bea Sys.	3	9,000	12	108,000	81,000
<b>Subtotal</b>						<b>\$109,150</b>	<b>\$82,757</b>
<b>User Connectivity</b>							
(8+1) port 10Mbps Ethernet Hub, 538 spares	NX-H9+	Netlux	4	47	5918	278,146	
HP Advancestack 2000 10Base-T module	J3102A		1	912	3	2,736	2,334
HP Advancestack 2000 FDDI module	J3108A		1	4,014	1	4,014	2,333
Advancestack Switch 2000	J3100B		1	1,824	1	1,824	1,555
<b>Subtotal</b>						<b>\$286,720</b>	<b>\$6,222</b>
<b>Total</b>						<b>\$3,419,788</b>	<b>\$810,106</b>
<b>Notes:</b> 1=Sun Data Corporation, 2=Sybase, 3=BEA Systems, 4=Netlux				<b>Five Year Cost: \$4,229,894</b>			
				<b>tpmC Rating: 52,117.80</b>			
				<b>\$/tpmC: \$81.17</b>			

Prices used in TPC benchmarks reflect the actual prices a customer would pay for a one-time purchase of the stated components. Individually negotiated discounts are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details, see the pricing sections of the TPC benchmark specifications. If you find that the stated prices are not available according to these terms, please inform the TPC at <pricing@tpc.org>. Thank you.

Note: audited by Richard Gimarc of Performance Metrics, Inc.

## Numerical Quantities Summary for HP 9000 V2250 Enterprise Server

**MQTH, Computed Maximum Qualified Throughput** **52,117.80 tpmC®**

<b>Response Times (in seconds)</b>	<b>90th %-ile</b>	<b>Maximum</b>	<b>Average</b>
New-Order	2.31	10.36	1.13
Payment	2.23	10.07	1.05
Order-Status	2.31	10.15	1.16
Delivery (interactive portion)	0.24	4.92	0.14
Delivery (deferred portion)	2.62	10.55	1.46
Stock-Level	3.35	10.13	1.73
Menu	0.1	7.40	0.02

### **Transaction Mix, in percent of total transactions**

New-Order	44.84%
Payment	43.02%
Order-Status	4.04%
Delivery	4.05%
Stock-Level	4.05%

### **Keying/Think Times (in seconds)**

	<b>Keying Time</b>			<b>Think Time</b>		
	<b>Min.</b>	<b>Avg.</b>	<b>Max.</b>	<b>Min.</b>	<b>Avg.</b>	<b>Max.</b>
New-Order	18.02	18.03	18.05	0.02	12.15	202.22
Payment	3.02	3.03	3.04	0.02	12.07	186.36
Order-Status	2.02	2.03	2.04	0.02	10.10	130.92
Delivery (interactive)	2.02	2.03	2.04	0.02	5.06	82.91
Stock-Level	2.02	2.03	2.04	0.02	5.06	57.26

**Numerical Quantities Summary for  
HP 9000 V2250 Enterprise Server, continued**

**Test Duration**

Ramp-up time	30.25 minutes
Measurement interval	30 minutes
Transactions during measurement interval	3486738
Ramp down time	5 minutes

**Checkpointing**

Number of checkpoints in measurement interval	1
Checkpoint interval	30 minutes

**Reproducibility Run**

51,990.77 tpmC - 0.24%



## Preface

---

This is the full disclosure report for a benchmark test of the HP 9000 V2250 Enterprise Server using Sybase Adaptive Server Enterprise 11.5. It meets the requirements of the TPC Benchmark<sup>®</sup> C Standard Specification, Revision 3.3.2 dated 25 June, 1997.

TPC Benchmark<sup>®</sup> C was developed by the Transaction Processing Performance Council (TPC). It is the intent of this group to develop a suite of benchmarks to measure the performance of computer systems executing a wide range of applications. Hewlett-Packard Company and Sybase, Inc. are active participants in the TPC.

### TPC Benchmark<sup>®</sup> C Overview

*TPC Benchmark<sup>®</sup> C is an On Line Transaction Processing (OLTP) workload. It is a mixture of read-only and update intensive transactions that simulate the activities found in complex OLTP application environments. It does so by exercising a breadth of system components associated with such environments, which are characterized by:*

- *The simultaneous execution of multiple transaction types that span a breadth of complexity*
- *On-line and deferred transaction execution modes*
- *Multiple on-line terminal sessions*
- *Moderate system and application execution time*
- *Significant disk input/output*
- *Transaction integrity (ACID properties)*
- *Non-uniform distribution of data access through primary and secondary keys*

- 
- *Databases consisting of many tables with a wide variety of sizes, attributes, and relationships*
  - *Contention of data access and update*

*The performance metric reported by TPC-C<sup>®</sup> is a “business throughput” measuring the number of orders processed per minute. Multiple transactions are used to simulate the business activity of processing an order, and each transaction is subject to a response time constraint. The performance metric for this benchmark is expressed in transactions-per-minute-C<sup>®</sup> (tpmC<sup>®</sup>). To be compliant with the TPC-C<sup>®</sup> standard, all references to tpmC<sup>®</sup> results must include the tpmC<sup>®</sup> rate, the associated price-per-tpmC<sup>®</sup>, and the availability date of the priced configuration.*

*Despite the fact that this benchmark offers a rich environment that emulates many OLTP applications, this benchmark does not reflect the entire range of OLTP requirements. In addition, the extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-C<sup>®</sup> approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to other environments are not recommended.*

Hewlett-Packard Company does not warrant or represent that a user can or will achieve performance similar to the benchmark results contained in this report. No warranty of system performance or price/performance is expressed or implied by this report.

---

# Table of Contents

---

<b>Abstract .....</b>	<b>iii</b>
<b>Table of Contents .....</b>	<b>xi</b>
<b>1 General Items .....</b>	<b>1-1</b>
1.1 Application Code and Definition Statements .....	1-1
1.2 Test Sponsor .....	1-1
1.3 Parameter Settings .....	1-1
1.4 Configuration Diagrams .....	1-2
<b>2 Clause 1 Related Items .....</b>	<b>2-1</b>
2.1 Table Definitions .....	2-1
2.2 Physical Organization of Database .....	2-1
2.3 Insert and Delete Operations .....	2-1
2.4 Partitioning .....	2-1
<b>3 Clause 2 Related Items .....</b>	<b>3-1</b>
3.1 Random Number Generation .....	3-1
3.2 Input/Output Screen Layout .....	3-1
3.3 Priced Terminal Feature Verification .....	3-1
3.4 Presentation Manager or Intelligent Terminal .....	3-2
3.5 Transaction Statistics .....	3-3
3.6 Queueing Mechanism .....	3-3
<b>4 Clause 3 Related Items .....</b>	<b>4-1</b>
4.1 Transaction System Properties (ACID) .....	4-1
4.2 Atomicity .....	4-1
4.2.1 Completed Transaction .....	4-1
4.2.2 Aborted Transaction .....	4-2
4.3 Consistency .....	4-2
4.4 Isolation .....	4-3
4.4.1 Isolation Test 1 .....	4-4
4.4.2 Isolation Test 2 .....	4-4
4.4.3 Isolation Test 3 .....	4-5
4.4.4 Isolation Test 4 .....	4-5
4.4.5 Isolation Test 5 .....	4-6
4.4.6 Isolation Test 6 .....	4-6
4.4.7 Isolation Test 7 .....	4-7
4.4.8 Isolation Test 8 .....	4-7
4.4.9 Isolation Test 9 .....	4-8
4.5 Durability .....	4-8
4.5.1 Durable Media Failure .....	4-9
4.5.2 Instantaneous Interruption and Loss of Memory .....	4-9
<b>5 Clause 4 Related Items .....</b>	<b>5-1</b>
5.1 Initial Cardinality of Tables .....	5-1
5.2 Database and Growth Layout .....	5-2
5.2.1 Data Model	

& Interfaces .....	5-4
5.2.2 Partitions / Replications .....	5-4
5.2.3 Growth Requirements .....	5-4
<b>6 Clause 5 Related Items .....</b>	<b>6-1</b>
6.1 Throughput .....	6-1
6.2 Response Times .....	6-1
6.3 Keying and Think Times .....	6-2
6.4 Response Time Frequency Distribution Curves and Other Graphs .....	6-2
6.5 Steady State Determination .....	6-7
6.6 Work Performed During Steady State .....	6-7
6.6.1 Checkpoint .....	6-7
6.6.2 Checkpoint Conditions .....	6-7
6.6.3 Checkpoint Implementation .....	6-7
6.7 Reproducibility .....	6-7
6.8 Measurement Period Duration .....	6-7
6.9 Regulation of Transaction Mix .....	6-7
6.10 Transaction Mix .....	6-8
6.11 Transaction Statistics .....	6-8
6.12 Checkpoint Count and Location .....	6-9
<b>7 Clause 6 Related Items .....</b>	<b>7-1</b>
7.1 RTE Description .....	7-1
7.2 Emulated Components .....	7-2
7.3 Functional Diagrams .....	7-2
7.4 Networks .....	7-2
<b>8 Clause 7 Related Items .....</b>	<b>8-1</b>
8.1 System Pricing .....	8-1
8.2 Support Pricing .....	8-1
8.2.1 HP Hardware Support .....	8-2
8.2.2 HP Software Support .....	8-2
8.2.3 Hubs .....	8-2
8.2.4 Sybase Standard Technical Support .....	8-2
8.3 Discounts .....	8-2
8.4 Availability .....	8-2
8.5 Priced System Configuration .....	8-2
8.6 Throughput, Price/Performance, and Availability Date .....	8-3
<b>9 Clause 9 Related Items .....</b>	<b>9-1</b>
9.1 Auditor's Report .....	9-1
<b>10 Report Availability .....</b>	<b>10-1</b>
<b>Appendix A: Client Source .....</b>	<b>A-1</b>
<b>Appendix B: Database Design .....</b>	<b>B-1</b>
<b>Appendix C: Tunable Parameters .....</b>	<b>C-1</b>
<b>Appendix D: RTE Configuration .....</b>	<b>D-1</b>
<b>Appendix E: Disk Storage .....</b>	<b>E-1</b>
<b>Appendix F: Price Quotes .....</b>	<b>F-1</b>

# 1 General Items

---

## 1.1 Application Code and Definition Statements

*The application program (as defined in clause 2.1.7) must be disclosed. This includes, but is not limited to, the code implementing the five transactions and the terminal input output functions.*

Appendix A contains the HP C/HP-UX application code used in this TPC-C<sup>®</sup> test.

## 1.2 Test Sponsor

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

The High Performance Systems Division of Hewlett-Packard Company is the test sponsor of this TPC Benchmark C<sup>®</sup>.

## 1.3 Parameter Settings

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

- *Database options,*
- *Recover/commit options,*
- *Consistency/locking options,*
- *Operating system and application configuration parameter.*
- *Compilation and linkage options and run-time optimizations used to create/install applications, OS, and/or databases.*

---

*This requirement can be satisfied by providing a full list of all parameters and options.*

*The intent of the above clause is that anyone attempting to recreate the benchmark environment has sufficient information to compile, link, optimize, and execute all software used to produce the disclosed benchmark result.*

Appendix A contains the application “make” files. Appendix C contains the HP-UX operating system parameters used to generate the kernel for the configuration used in this benchmark. Also included are all of the Sybase Adaptive Server Enterprise 11.5 database parameters and the TUXEDO transaction monitor parameters used.

## **1.4 Configuration Diagrams**

*Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences.*

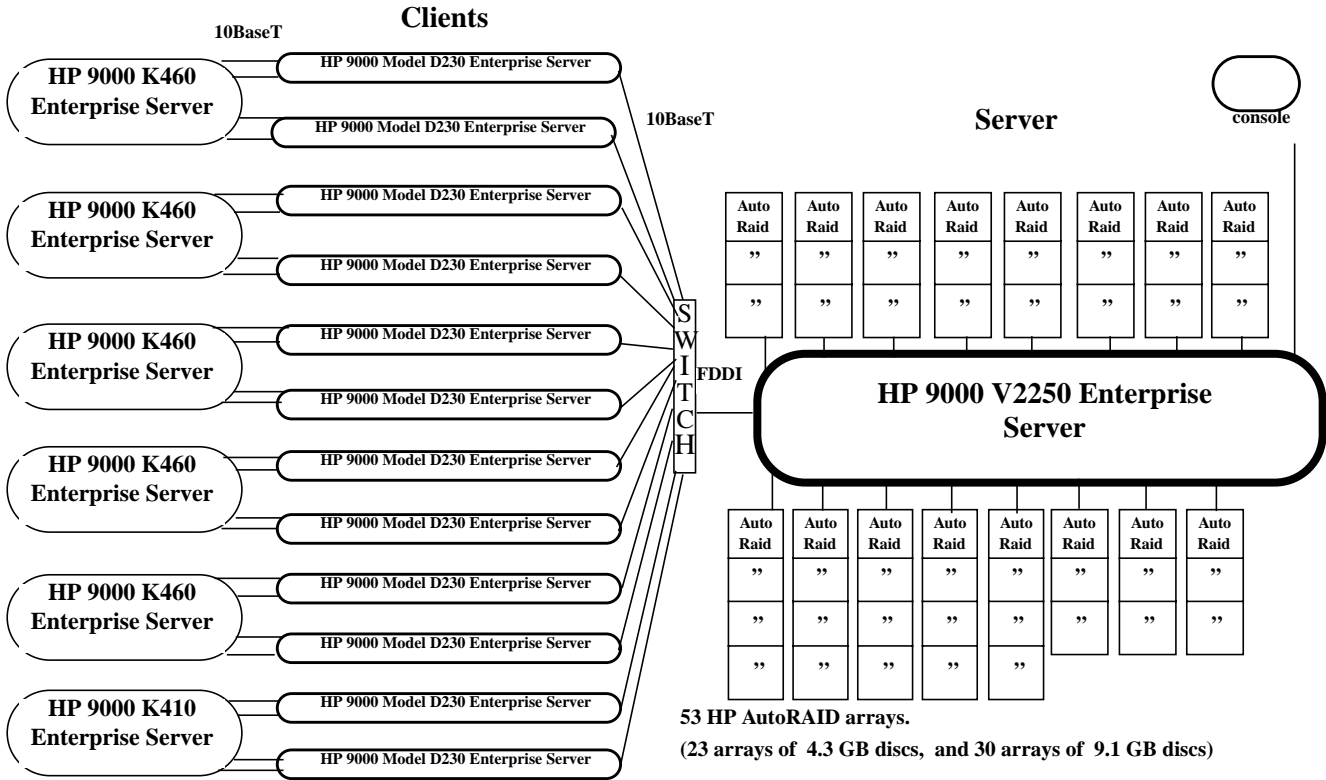
The server System Under Test, an HP 9000 V2250 Enterprise Server depicted in Figure 1.1, consisted of:

- 16 240 MHz PA-RISC 8200 System Processors
- 16 GB of memory
- 16 HP-PCI Fast/Wide SCSI interfaces
- 53 HP AutoRAID Arrays
- One FDDI LAN interface

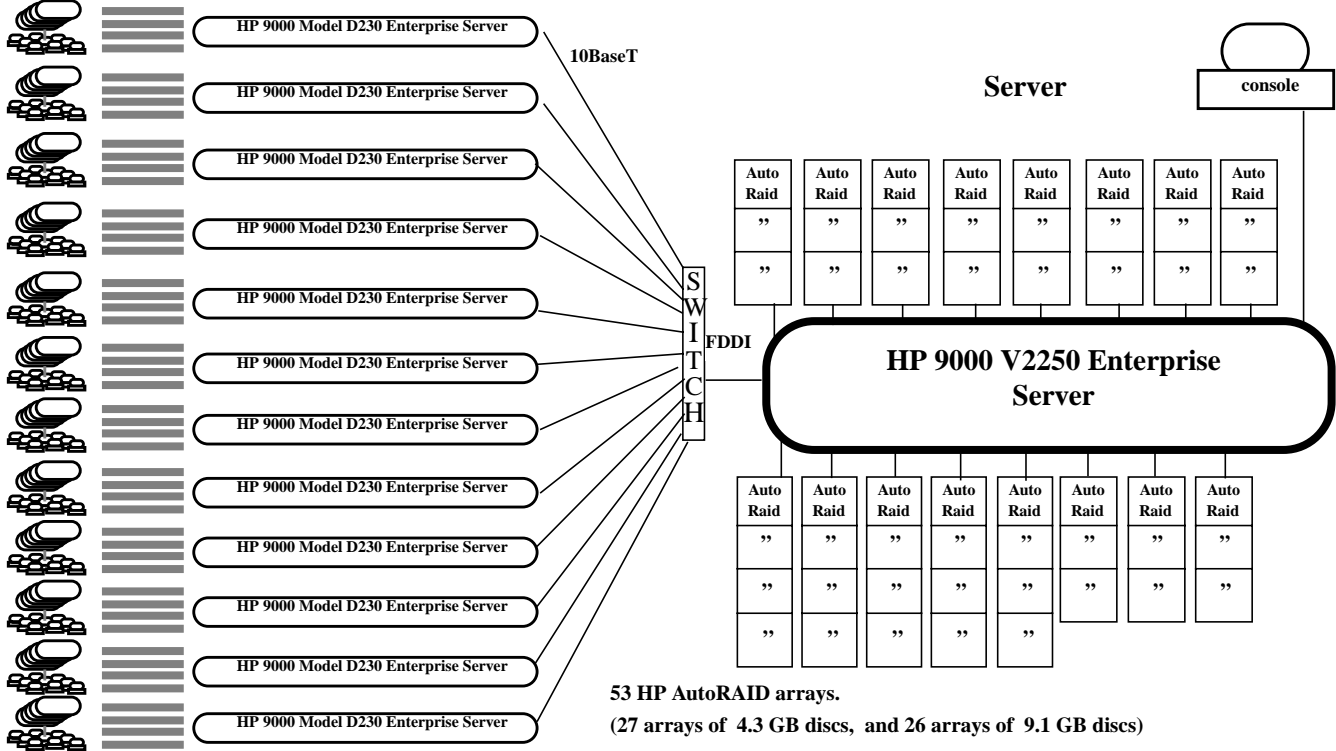
As indicated in Figure 1.1, this benchmark configuration used Remote Terminal Emulator (RTE) programs that executed on HP9000 Model K460 (five) and K410 (one) Enterprise Server drivers to emulate TPC-C<sup>®</sup> user sessions. The emulated users on the driver system were directly connected to the client systems under test via twenty-four separate 10BaseT local area network (LAN) and communicated using TCP/IP. The clients were connected to the SUT via a Ethernet/FDDI switch.

The priced configuration for the a HP 9000 V2250 Enterprise Server is shown in Figure 1.2. In the priced configuration, the RTE shown in the benchmark configuration is replaced by the appropriate number of workstations (emulating ANSI terminals) connected to Netlux hubs.

**Figure 1.1: HP 9000 V2250 Enterprise Server Benchmark Configuration**



**Figure 1.2: HP 9000 V2250 Enterprise Server Priced Configuration**







## 2 Clause 1 Related Items

---

### 2.1 Table Definitions

*Listing must be provided for all table definition statements and all other statements used to set up the database.*

Appendix B describes the programs that define, create, and populate an Sybase database for TPC-C<sup>®</sup> testing.

### 2.2 Physical Organization of Database

*The physical organization of tables and indices, within the database, must be disclosed.*

Space was allocated to Sybase Adaptive Server Enterprise 11.5 according to the data in section 5.2. The size of the database table space on each disk drive was calculated to provide even distribution of load across the disk drives.

### 2.3 Insert and Delete Operations

*It must be ascertained that insert and/or delete operations to any of the tables can occur concurrently with the TPC-C<sup>®</sup> transaction mix. Furthermore, any restrictions in the SUT database implementation that precludes inserts beyond the limits defined in Clause 1.4.11 must be disclosed. This includes the maximum number of rows that can be inserted and the maximum key value for these new rows.*

### 2.4 Partitioning

There were no restrictions on insert and delete operations to any tables.

*While there are a few restrictions placed upon horizontal or vertical partitioning of tables and rows in the TPC-C<sup>®</sup> benchmark, any such*

---

*partitioning must be disclosed. Replication of tables, if used, must be disclosed. Additional and/or duplicated attributes in any table must be disclosed along with a statement on the impact on performance.*

Partitioning, replication, and additional or duplicated attributes were not used in this implementation.

## 3 *Clause 2 Related Items*

---

- 3.1 Random Number Generation** *The method of verification for the random number generation must be disclosed.*
- The random number generator used can be found in the source appendix. It is from the book “The Art of Computer Systems Performance Analysis” by Raj Jain, page 443. The properties of this random number generator are documented in the book. It is a full-period multiplicative linear-congruential random number generator.
- 3.2 Input/Output Screen Layout** *The actual layout of the terminal input/output screens must be disclosed.*
- The screen layouts corresponded exactly to those in Clauses 2.4.3, 2.5.3, 2.6.3, 2.7.3, and 2.8.3 of the TPC-C<sup>®</sup> Standard Specification.
- 3.3 Priced Terminal Feature Verification** *The method used to verify that the emulated terminals provide all the features described in Clause 2.2.2.4 must be explained. Although not specifically priced, the type and model of the terminals used for the demonstration in 8.1.3.3 must be disclosed and commercially available (including supporting software and maintenance).*

The terminal features were verified by manually exercising each specification on an HP 712/80 workstation running an ANSI terminal emulator.

**3.4 Presentation Manager or Intelligent Terminal** *Any usage of presentation managers or intelligent terminals must be explained.*

Application code running on the client implemented the TPC-C user interface. A listing of this code is included in Appendix A. Used capabilities of the terminal beyond basic ASCII entry and display were restricted to cursor positioning.

A presentation manager was not used.

**Table 3.1: Transaction Statistics**

Statistics		Value
New Order	Home warehouse items	99.00%
	Remote warehouse items	1.00%
	Rolled back transactions	0.99%
	Average items per order	10.00
Payment	Home warehouse	85.01%
	Remote warehouse	14.99%
	Non-primary key access	60.01%
Order Status	Non-primary key access	60.05%
Delivery	Skipped transactions	0
Transaction Mix	New order	44.84%
	Payment	43.02%
	Order status	4.04%
	Delivery	4.05%
	Stock level	4.05%

---

**3.5 Transaction Statistics** Table 3.1 lists the numerical quantities that Clauses 8.1.3.5 to 8.1.3.11 require.

**3.6 Queueing Mechanism** The queueing mechanism used to defer the execution of the Delivery transaction must be disclosed.

Delivery transactions were submitted to servers using the same TUXEDO mechanism that other transactions used. The only difference was that the call was asynchronous, i.e., control would return to the client process immediately and the deferred delivery part would complete asynchronously.



## 4 Clause 3 Related Items

---

- 4.1 Transaction System Properties (ACID)** *The results of the ACID tests must be disclosed along with a description of how the ACID requirements were met. This includes disclosing which case was followed for the execution of Isolation Test 7.*

The TPC Benchmark C Standard Specification defines a set of transaction processing system properties that a system under test (SUT) must support during the execution of the benchmark. Those properties are Atomicity, Consistency, Isolation, and Durability (ACID). This section quotes the specification definition of each of these properties and describes the tests done as specified and monitored by the auditors, to demonstrate compliance.

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

- 4.2.1 Completed Transaction** *Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number as specified in Clause 2.5.1.2) and verify that the records in the CUSTOMER, WAREHOUSE, and DISTRICT tables have been changed appropriately.*

The values of w\_ytd, d\_ytd, c\_balance, c\_ytd\_payment, and c\_payment\_cnt of a randomly selected warehouse, district, and customer were retrieved. The Payment transaction was executed on the same warehouse, district, and customer. The transaction was committed. The values w\_ytd, d\_ytd, c\_balance, c\_ytd\_payment, and c\_payment\_cnt were

---

retrieved again. It was verified that all values had been changed appropriately.

### 4.2.2 Aborted Transaction

*Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number as specified in Clause 2.5.1.2) and substitute a ROLLBACK of the transaction for the COMMIT of the transaction. Verify that the records in the CUSTOMER, WAREHOUSE, and DISTRICT tables have NOT been changed*

The values of w\_ytd, d\_ytd, c\_balance, c\_ytd\_payment and c\_payment\_cnt of a randomly selected warehouse, district, and customer were retrieved. The Payment transaction was executed on the same warehouse, district, and customer. The transaction was rolled back. The values of w\_ytd, d\_ytd, c\_balance, c\_ytd\_payment, c\_payment\_cnt were retrieved again. It was verified that none of the values had changed.

## 4.3 Consistency

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

The TPC Benchmark C standard requires the System Under Test to meet the following 12 consistency conditions (c.f. 3.3.2.1 to 3.3.2.12):

1. the sum of the district balances in a warehouse is equal to the warehouse balance;
2. for each district, the next order-id minus one is equal to maximum order-id in the ORDER table and equal to the maximum new order-id in the NEW ORDER table;
3. for each district, the maximum order-id minus minimum order-id in the ORDER table plus one equals the number of rows in the NEW-ORDER table for that district;
4. for each district, the sum of the order-line counts equals the number of rows in the ORDER-LINE table for that district;
5. for each row in the ORDER table, the carrier-id is set to a null value only if there is a corresponding row in the NEW ORDER table;
6. for each row in the ORDER table, the order-line count must equal the number of rows in the ORDER-LINE table for that order;
7. for any row in the ORDER-LINE table, the delivery date/time is set to a null value only if the corresponding row in the ORDER table has the carrier-id set to a null value;
8. for each warehouse, the year-to-date amount must equal the sum of the amounts in the HISTORY table for that warehouse;
9. for each district, the year-to-date amount must equal the sum of the amounts in the HISTORY table for that district;



- 
10. for each customer, the balance must equal the sum of the order-line amount minus the sum of the history amount for that customer;
  11. for each district, the total orders minus the total new-orders must equal the sum of the customer delivery count;
  12. for any randomly selected customer, the balance plus the year-to-date payment must equal the sum of the order-line amount.

The TPC Benchmark C Standard Specification requires explicit demonstration that the conditions are satisfied for the first four conditions only.

In order to demonstrate the consistency of the application, the following steps were taken:

1. The consistency of the database was verified by applying the conditions 1-4 described above on a freshly populated database.
2. A test was performed under the normal user load at steady state and containing one checkpoint.
3. Upon the completion of the test, the consistency of the database was determined by applying the same consistency conditions used in step 1.

## 4.4 Isolation

*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 arbitrary mix of TPC-C transactions, unless otherwise specified by the transaction profile. The system or application must have full serializability enabled (i.e., repeated reads of the same rows within any committed transaction must return identical data when run concurrently with any arbitrary mix of TPC-C transactions), except in the case of Stock-Level transaction. For the Stock-Level transaction, the isolation requirement is relaxed to simply require that the transaction see only committed data.*

The TPC Benchmark C Standard defines seven required tests to be performed to demonstrate that the required levels of transaction isolation are met. All seven required tests were performed successfully. In addition to those seven tests, three more tests specified by the auditor were performed successfully. These additional tests demonstrated phantom protection within any mix of TPC-C transactions.

*For conventional locking schemes, isolation should be tested as described below. Systems that implement other isolation schemes may require differ-*

---

*ent validation techniques. It is the responsibility of the test sponsor to disclose those techniques and the tests for them. If isolation schemes other than conventional locking are used, it is permissible to implement these tests differently provided full details are disclosed. (Examples of different validation techniques are shown in Isolation Test 7, Clause 3.4.2.7).*

#### **4.4.1 Isolation Test 1**

*This test demonstrates isolation for read-write conflicts of Order-Status and New-Order transactions.*

The execution of the above test proceeded as follows:

1. An Order-Status transaction T0 was executed for a randomly selected customer, and the order returned was noted. T0 was committed
2. A New-Order transaction T1 was started for the same customer used in T0. T1 was stopped prior to COMMIT.
3. An Order-Status transaction T2 was started for the same customer used in T1. T2 completed and was committed without being blocked by T1. T2 returned the same order that T0 had returned.
4. T1 was allowed to complete and was committed.
5. An Order-Status transaction T3 was started for the same customer used in T1. T3 returned the order inserted by T1.

This outcome demonstrates serialization of T2 before T1. It has equivalent validity to the outcome specified in the Standard which supposes T1 to be serialized before T2.

#### **4.4.2 Isolation Test 2**

*This test demonstrates isolation for read-write conflicts of Order-Status and New-Order transactions when the New-Order transaction is ROLLED BACK.*

The execution of the above test proceeded as follows:

1. An Order-Status transaction T0 was executed for a randomly selected customer and the order returned was noted. T0 was committed.
2. A New-Order transaction T1 with an invalid item number, was started for the same customer used in T0. T1 was stopped immediately prior to ROLLBACK.
3. An Order-Status transaction T2 was started for the same customer used in T1. T2 completed and was committed without being blocked by T1. T2 returned the same order that T0 had returned.
4. T1 was allowed to ROLLBACK.
5. An Order-Status transaction T3 was started for the same customer used in T1. T3 returned the same order that T0 had

---

returned.

### 4.4.3 Isolation Test 3

*This test demonstrates isolation for write-write conflicts of two New-Order transactions.*

The execution of the above test proceeded as follows:

1. Started a New-Order transaction T1.
1. Stopped transaction T1 immediately prior to COMMIT.
2. Another New-Order transaction T2 was started for the same customer used in T1. T2 waited.
3. T1 was allowed to complete. T2 completed and was committed.
4. Transaction T1 was allowed to complete. T2 also completed.
5. Verified the order number returned for T2 was one greater than the order number for T1. Also the value of D\_NEXT\_O\_ID reflected the results for both T1 and T2.

### 4.4.4 Isolation Test 4

*This test demonstrates isolation for write-write conflicts of two New-Order transactions when one transaction is ROLLED BACK.*

The execution of the above test proceeded as follows:

1. The D\_NEXT\_O\_ID of a randomly selected district was retrieved.
2. A New-Order transaction T1, with an invalid item number, was started for a randomly selected customer within the district used in step 1. T1 was stopped immediately prior to ROLLBACK.
3. Another New-Order transaction T2 was started for the same customer used in T1. T2 waited.
4. T1 was allowed to roll back, and T2 completed and was committed.
5. The order number returned by T2 was the same as the D\_NEXT\_O\_ID retrieved in step 1.
6. The D\_NEXT\_O\_ID of the same district was retrieved again. It had been incremented by one (i.e. one greater than the order number returned by T2).

---

#### 4.4.5 Isolation Test 5

*This test demonstrates isolation for write-write conflicts of Payment and Delivery transactions.*

The execution of the above test proceeded as follows:

1. A query was executed to find out the customer who would be updated by the next delivery transaction for a randomly selected warehouse and district.
2. The C\_BALANCE of the customer found in step 1 was retrieved.
3. A Delivery business transaction T1 was started for the same warehouse used in step 1. T1 was stopped immediately prior to the COMMIT of the database transaction corresponding to the district used in step 1.
4. A Payment transaction T2 was started for the same customer found in step 1. T2 waited.
5. T1 was allowed to complete. T2 completed and was committed.
6. The C\_BALANCE of the customer found in step 1 was retrieved again. The C\_BALANCE reflected the results of both T1 and T2.

#### 4.4.6 Isolation Test 6

*This test demonstrates isolation for write-write conflicts of Payment and Delivery transactions when the Delivery transaction is ROLLED BACK.*

The execution of the above test proceeded as follows:

1. A query was executed to find out the customer who would be updated by the next delivery transaction for a randomly selected warehouse and district.
2. The C\_BALANCE of the customer found in step 1 was retrieved.
3. A Delivery business transaction T1 was started for the same warehouse used in step 1. T1 was stopped immediately prior to the ROLLBACK of the database transaction corresponding to the district used in step 1.
4. A Payment transaction T2 was started for the same customer found in step 1. T2 waited.
5. T1 was allowed to ROLLBACK. T2 completed and was committed.
6. The C\_BALANCE of the customer found in step 1 was retrieved again. The C\_BALANCE reflected the results of only T2.

---

#### 4.4.7 Isolation Test 7

*This test demonstrates repeatable reads for the New-Order transaction while an interactive transaction updates the price of an item.*

The execution of the above test proceeded as follows:

1. The I\_PRICE of two randomly selected items X and Y were retrieved.
2. A New-Order transaction T2 with a group of items including items X and Y was started. T2 was stopped immediately after retrieving the prices of all items. The prices of items X and Y retrieved matched those retrieved in step 1.
3. A transaction T3 was started to increase the price of items X and Y by 10%.
4. T3 did not stall and no transaction was rolled back. T3 was committed.
5. T2 was resumed, and the prices of all items were retrieved again within T2. The prices of items X and Y matched those retrieved in step 1.
6. T2 was committed.
7. The prices of items X and Y were retrieved again. The values matched the values set by T3.

- Case D was followed in this isolation test

Three additional TPC-approved Isolation tests were added by the auditors to validate the requirement of *Clause 3.4.1* that there be *Level 3* (“No Phantoms”) *Isolation between New-Order, Payment, Delivery and Order-Status Transactions*.

#### 4.4.8 Isolation Test 8

*This test demonstrates isolation for phantom protection between New-Order and Order-Status transactions.*

The execution of the above test proceeded as follows:

1. An Order-Status transaction T1 was started for a randomly selected customer.
2. T1 was stopped immediately after reading the order table for the selected customer. The most recent order for that customer was found.
3. A New-Order transaction T2 was started for the same customer. T2 completed and was committed without being blocked by T1.
4. T1 was resumed and the order table was read again to determine the most recent order for the same customer. The order

- 
- found was the same as the one found in step 2.
  5. T1 completed and was committed.

#### 4.4.9 Isolation Test 9

*This test demonstrates isolation for phantom protection between New-Order and Delivery transactions.*

The execution of the above test proceeded as follows:

1. The NO\_D\_ID of all new\_order rows for a randomly selected warehouse and district was changed to 11. The changes were committed.
2. A Delivery transaction T1 was started for the selected warehouse.
3. T1 was stopped immediately after reading the new\_order table for the selected warehouse and district. No qualifying row was found.
4. A New-Order transaction T2 was started for the same warehouse and district. T2 completed and was committed without being blocked by T1.
5. T1 was resumed and the new\_order table was read again. No qualifying row was found.
6. T1 completed and was committed.
7. The NO\_D\_ID of all new\_order rows for the selected warehouse and district was restored to the original value. The changes were committed.

#### 4.5 Durability

*The tested system must guarantee durability: the ability to preserve the effects of committed transaction and insure database consistency after recovery from any one of the failures listed in Clause 3.5.3.*

*List of single failures:*

*Permanent irrecoverable failure of any single durable medium containing TPC-C database tables or recovery log data.*

*Instantaneous interruption (system crash/system hang) in processing which requires system reboot to recover.*

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

Hewlett-Packard executed specified durability tests to demonstrate satisfaction of the durability requirements for this implementation of TPC Benchmark C. All durability tests (including loss of memory / instantaneous interruption, loss of redo log disk, and loss of data disk) were per-

---

formed under full load of 43,000 terminals on a 4300 warehouse database (full scale).

#### 4.5.1 Durable Media Failure

The log and data-storage devices, Redundant Disk Arrays, function independently in ensuring data integrity under loss (or replacement) of any individual disk drive (and other failures) without interruption in processing. Therefore, the loss of a data disk and loss of a log disk were performed during the loss of system test.

#### 4.5.2 Instantaneous Interruption and Loss of Memory

Instantaneous interruption and loss of memory tests were combined because the loss of power erases the contents of memory. In addition, the loss of log and loss of disk were combined in this test because all data resides on RAID storage that does not cause data loss due to the loss of a single component. The system interruption and memory failure was induced by removing the SUT's primary power while the benchmark was running. This test was performed with a fully scaled SUT, fully scaled database under the full load of 43,000 terminals.

1. The D\_NEXT\_O\_ID fields for all rows in district table were summed up to determine the initial count of the total number of orders (count1).
2. A test was executed with 43,000 terminals. On the driver system, completed/rolled-back New-Order transactions were recorded as such in a "success" file.
3. After approximately five minutes at full load, a disk was pulled from the log array and a disk was pulled from a data array. Processing continued as normal due to the RAID redundancy.
4. After another 10 minutes at full load, the server system was unpowered by turning off the power on the front panel.
5. The test was aborted on the driver.
6. The pulled drives were re-inserted into the arrays and the arrays proceeded to "rebuild" the redundancy that was lost.
7. The server system was restarted.
8. The database was restarted and a recovery performed using the transaction log.
9. The contents of the "success" file on the driver and the ORDERS table were spot-compared to verify that records in the "success" file for completed New-Order transactions had corresponding records in the ORDERS table and no entries appeared for rolled-back transactions.
10. Step 1 was repeated to determine the current total number of orders (count2). Count2-count1 was three greater than the number of transactions recorded in the results log on the RTE. This inequality due to "in-flight" transactions (*committed on the system under test but for which the output data was not displayed on the [emulated] input/output screen before the failure*) was within allowed conditions.





## 5 Clause 4 Related Items

---

### 5.1 Initial Cardinality of Tables

*The cardinality (e.g. number of rows) of each table, as it existed at the start of the benchmark run, must be disclosed. If the database was overscaled and inactive rows of the WAREHOUSE table were deleted the cardinality of the WAREHOUSE table as initially configured and the number of rows deleted must be disclosed.*

The TPC-C database for this test was configured with 4,300 warehouses.

**Table 5.1: Number of Rows in Initial DB**

<b>Table</b>	<b>Occurrences</b>
Warehouse	4,300
District	43,000
Customer	129,000,000
History	129,000,000
Orders	129,000,000
New order	38,700,000
Order line	1,290,120,893
Stock	430,000,000
Item	100,000

## 5.2 Database and Growth Layout *The distribution of tables and logs across all media must be explicitly depicted for tested and priced systems.*

Table 5.2 indicates the distribution of the database tables over the disks of the tested and priced systems.

**Table 5.2: Disk Allocation**

Table Chunk	Disk Address	LVM Usage	Allocated Space (MB)
OS+SWAP	c0t6d0		2048
/project	c0t6d1		2048
SWAP	c1t15d2		16384
master	vgtpcc	12 way striping	250
master2	vgtpcc	12 way striping	50
tmpdb	vgtpcc	12 way striping	59
item, warehouse, district, new_order	vgtpcc	12 way striping	600
history	vgtpcc	12 way striping	8400
history2	vgtpcc	12 way striping	200
orders1	c8t14d1		345
orders2	c9t14d1		345
orders3	c9t15d1		345
orders4	c10t15d1		345
orders5	c12t13d1		345
orders6	c1t13d1		345
orders7	c20t13d1		345
orders8	c2t13d1		345
orders9	c3t13d1		345
orders10	c5t13d1		345
orders11	vgtpcc	12 way striping	800
orders12	vgtpcc	12 way striping	200
order_line1	c8t13d1		7735
order_line2	c2t12d1		7735
order_line3	c0t15d1		7735
order_line4	c8t15d1		7735
order_line5	c5t15d1		7735
order_line6	c3t15d1		7735
order_line7	c2t15d1		7735
order_line8	c20t15d1		7735
order_line9	c1t15d1		7735
order_line10	c12t15d1		7735
order_line11	vgtpcc	12 way striping	16500
order_line12	vgtpcc	12 way striping	800
customer1	c10t13d0		7380
customer2	c11t13d0		7380
customer3	c13t13d0		7380
customer4	c14t13d0		7380
customer5	c18t13d0		7380
customer6	c19t13d0		7380
customer7	c4t13d0		7380
customer8	c9t13d0		7380

**Table 5.2: Disk Allocation**

Table Chunk	Disk Address	LVM Usage	Allocated Space (MB)
customer9	c11t14d0		7380
customer10	c13t14d0		7380
customer11	c14t14d0		7380
customer12	c19t14d0		7380
customer13	vgtpec	12 way striping	4500
c_index1	c11t15d1		1024
c_index2	c13t15d1		1024
c_index3	c14t15d1		1024
c_index4	c18t15d1		1024
c_index5	c19t15d1		1024
c_index6	c20t14d1		1024
c_index7	c4t15d1		1500
c_index8	vgtpec	12 way striping	4000
stock1	c11t15d0		3930
stock2	c13t15d0		3930
stock3	c14t15d0		3930
stock4	c18t15d0		3930
stock5	c19t15d0		3930
stock6	c20t15d0		3930
stock7	c4t15d0		3930
stock8	c8t14d0		3930
stock9	c9t14d0		3930
stock10	c9t15d0		3930
stock11	c10t15d0		3930
stock12	c12t13d0		3930
stock13	c1t13d0		3930
stock14	c20t13d0		3930
stock15	c2t13d0		3930
stock16	c3t13d0		3930
stock17	c5t13d0		3930
stock18	c8t13d0		3930
stock19	c2t12d0		3930
stock20	c0t15d0		3930
stock21	c8t15d0		3930
stock22	c5t15d0		3930
stock23	c3t15d0		3930
stock24	c2t15d0		3930
stock25	c20t15d0		3930
stock26	c1t15d0		3930
stock27	c12t15d0		3930
stock28	c20t12d0		3930
stock29	c19t12d0		3930
stock30	c10t12d0		3930
stock31	c5t14d0		3930
stock32	c4t14d0		3930
stock33	c3t14d0		3930
stock34	c2t14d0		3930
stock35	c1t14d0		3930
stock36	c18t14d0		3930
stock37	c12t14d0		3930
stock38	c10t14d0		3930
stock39	c0t14d0		3930
stock40	vgtpec	12 way striping	7665
log1	c0t6d2		20000

The disks used in the 12-way LVM stripe were: c20t12d1, c19t12d1, c10t12d1, c5t14d1, c4t14d1, c3t14d1, c2t14d1, c1t14d1, c18t14d1, c12t14d1, c10t14d1, c0t14d1

---

The distribution of the database tables over the 53 disk arrays of the priced system is an extension of the distribution in the tested system. 180-day storage growth requirements are met with the unused space of the priced configuration. Figure 1.2 shows the configuration of the priced-system disks; capacity is indicated in Appendix E.

---

## 5.2.1 Data Model & Interfaces

*A statement must be provided that describes:*

1. *The data model implemented by the DBMS used (e.g. relational, network, hierarchical)*
2. *The database interface used (e.g. embedded, call-level) and access language (e.g. SQL, DL/1, COBOL, read/write) used to implement the TPC-C transactions. If more than one interface/access language is used to implement TPC-C, each interface/access language must be described and a list of which interface/access language is used with which transaction type must be disclosed.*

Sybase Adaptive Server Enterprise functions as a relational DBMS. SQL stored procedures were invoked through the Sybase Open Client DB-Library interface; the application code appears in Appendix A

## 5.2.2 Partitions / Replications

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

No partitioning or replication was used.

## 5.2.3 Growth Requirements

*Details of the 180 day space computations along with proof that the database is configured to sustain 8 hours of growth for the dynamic tables (Order, Order-Line, and History) must be disclosed.*

See Appendix E.

The database log and tables were validated as being able to accommodate 8 hours of growth at the measured throughput by extrapolating from the 60 minute full-load run containing the reported measurement interval. The growth over this run was extrapolated to 8 hours at the measured full load in proportion to number of New-Order transactions submitted, and the growth added to the size of the tables upon completion of the load of the initial population.



## 6 Clause 5 Related Items

---

### 6.1 Throughput

*Measured tpmC must be reported.*

tpmC®	52,117.80
-------	-----------

**Table 6.1: Measured tpmC**

### 6.2 Response Times

*Ninetieth percentile, maximum and average response times must be reported for all transaction types as well as for the menu response time.*

Type	Average	Maximum	90th percentile
New-Order	1.13	10.36	2.31
Payment	1.05	10.07	2.23
Order-Status	1.16	10.15	2.31
Interactive Delivery	0.14	4.92	0.24
Deferred Delivery	1.46	10.55	2.62
Stock-Level	1.73	10.13	3.35

**Table 6.2: Response Times**

Type	Average	Maximum	90th percentile
Menu	0.02	7.40	0.1

**Table 6.2: Response Times**

### 6.3 Keying and Think Times

*The minimum, the average, and the maximum keying and think times must be reported for each transaction type.*

Type	Minimum	Average	Maximum
New-Order	18.02	18.03	18.05
Payment	3.02	3.03	3.04
Order-Status	2.02	2.03	2.04
Interactive Delivery	2.02	2.03	2.04
Stock-Level	2.02	2.03	2.04

**Table 6.3: Keying Times**

Type	Minimum	Average	Maximum
New-Order	0.02	12.15	202.22
Payment	0.02	12.07	186.36
Order-Status	0.02	10.10	130.92
Interactive Delivery	0.02	5.06	82.91
Stock-Level	0.02	5.06	57.26

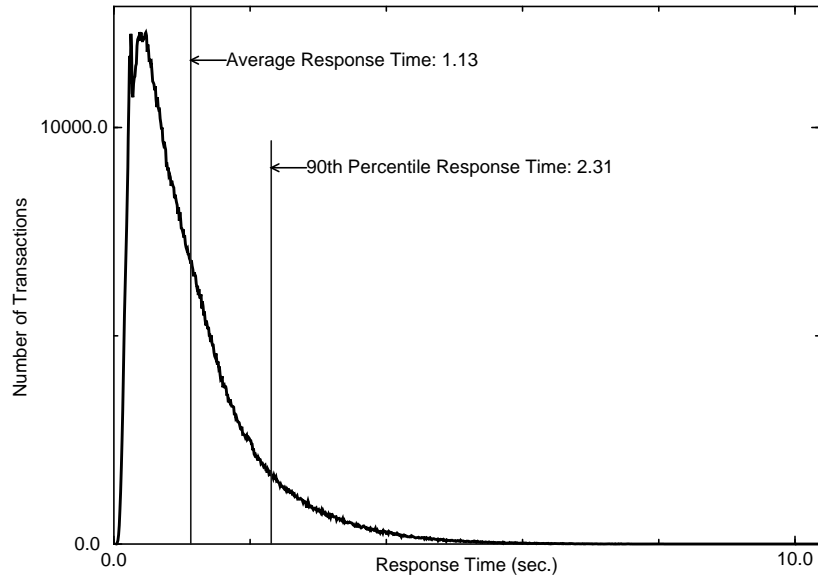
**Table 6.4: Think Times**

### 6.4 Response Time Frequency Distribution Curves and Other Graphs

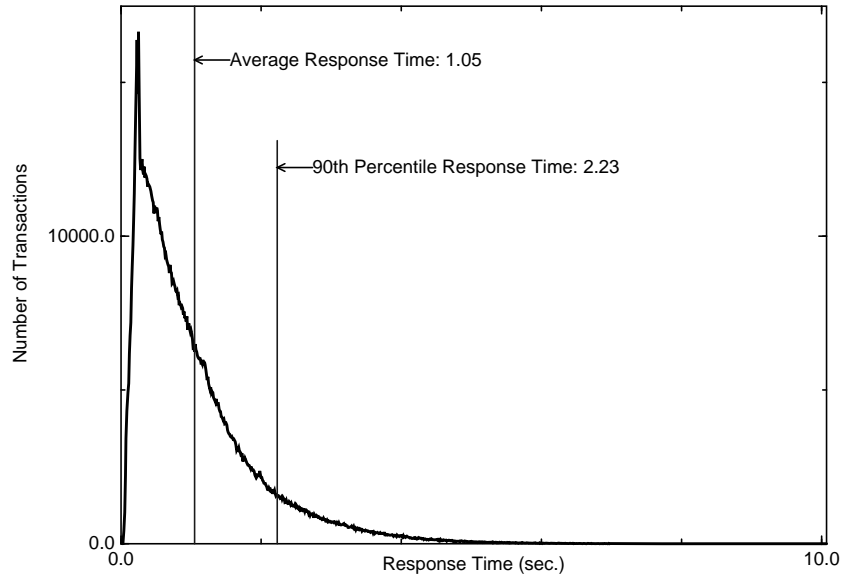
*Response Time frequency distribution curves (see Clause 5.6.1) must be reported for each transaction type. The performance curve for response times versus throughput (see Clause 5.6.2) must be reported for the New-Order transaction. The Think Time frequency distribution curve (see Clause 5.6.3) must be reported for the New-Order transaction. A graph of throughput versus elapsed time (see Clause 5.6.5) must be reported for the New-Order transaction, and the measurement interval indicated.*



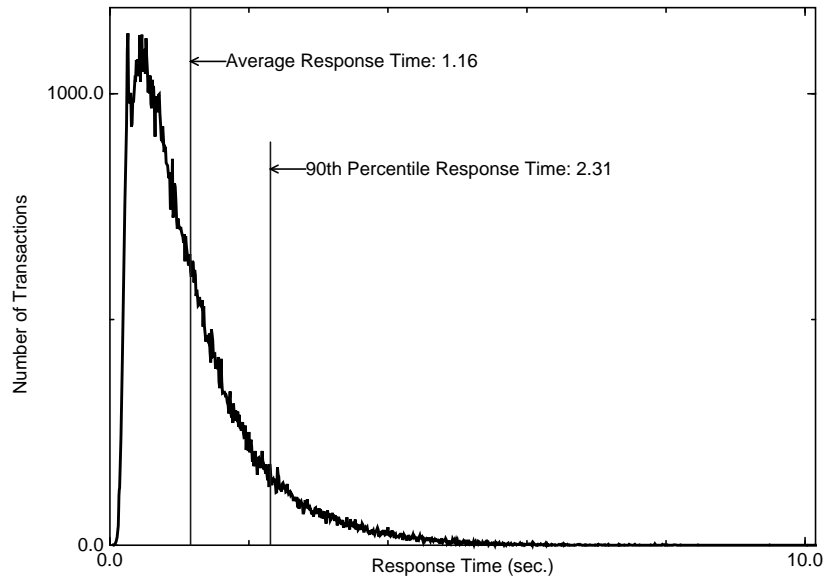
**Figure 6.1: New Order Response Time Distribution**



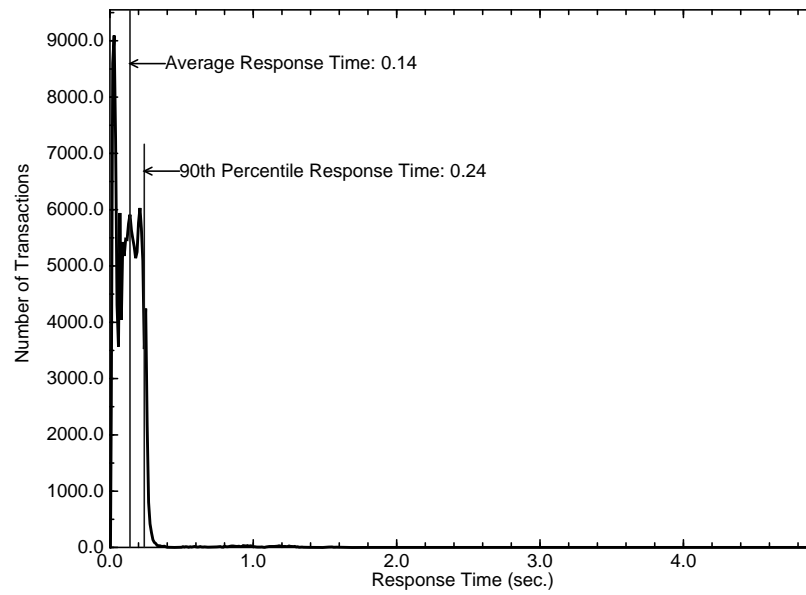
**Figure 6.2: Payment Response Time Distribution**



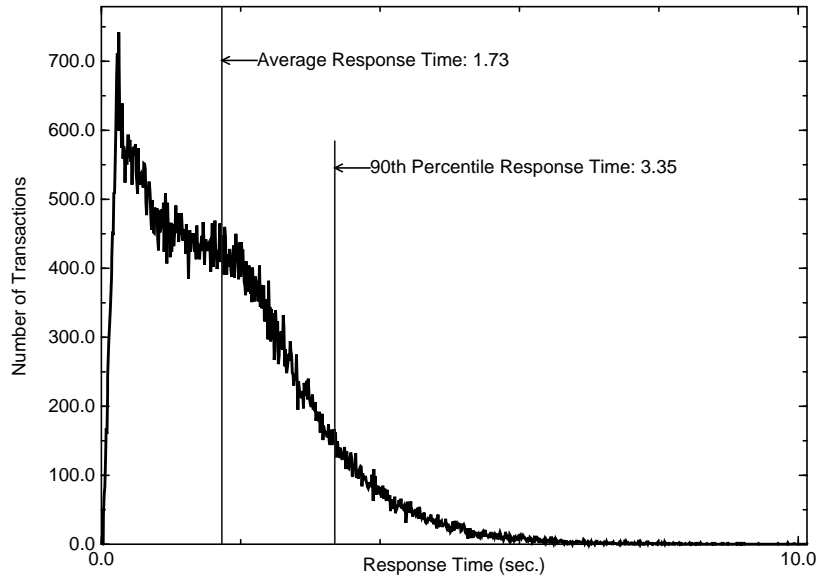
**Figure 6.3: Order Status Response Time Distribution**



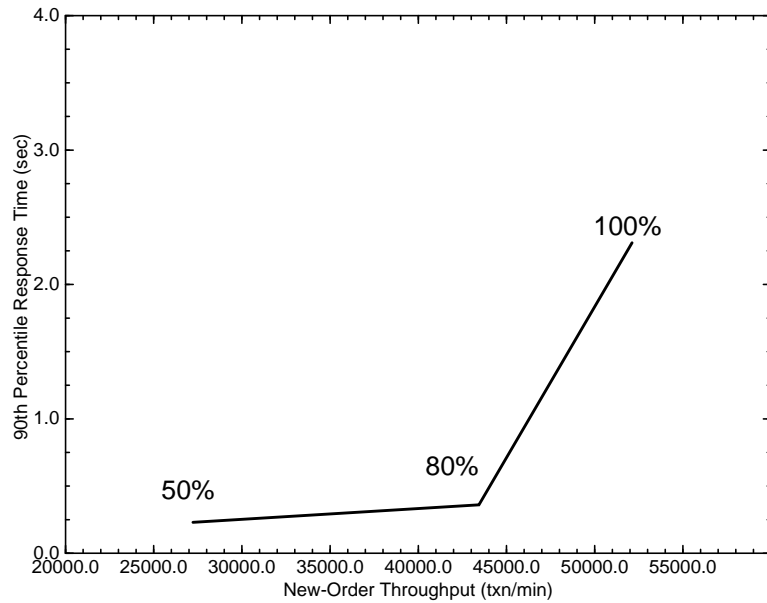
**Figure 6.4: Delivery Response Time Distribution**



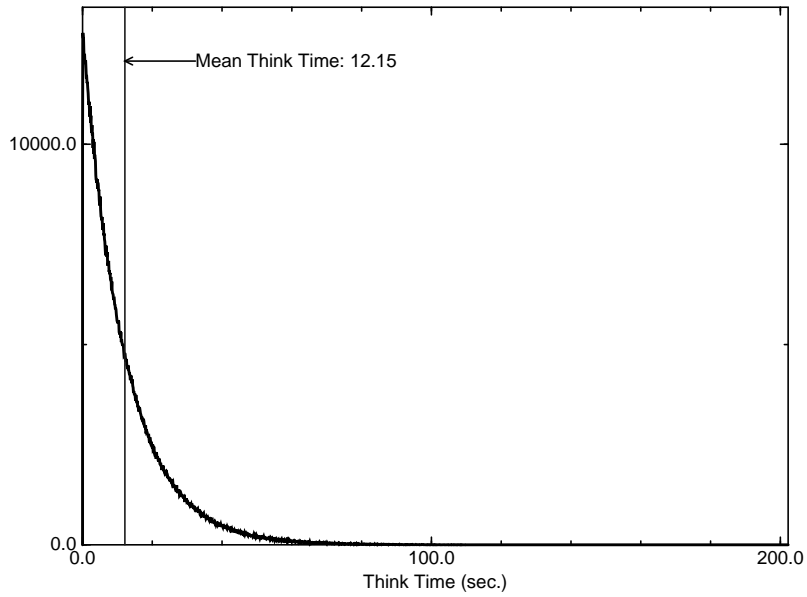
**Figure 6.5: Stock Level Response Time Distribution**



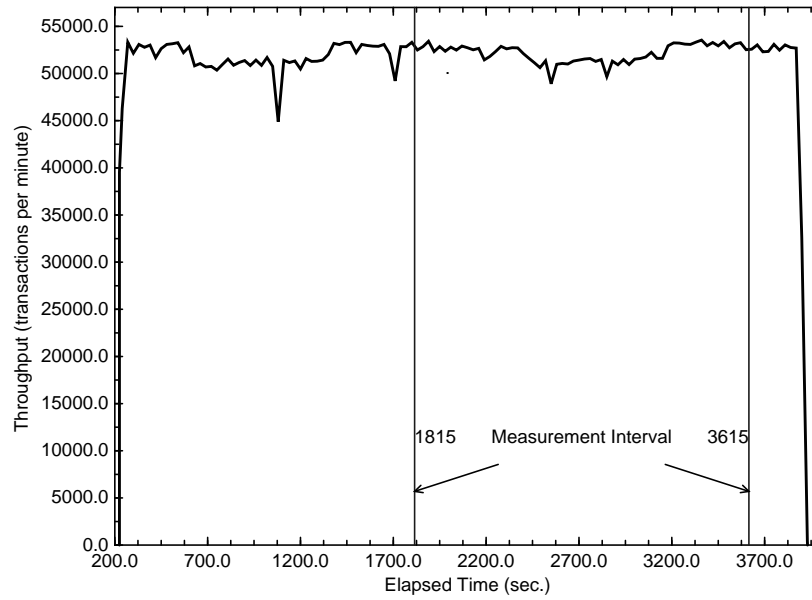
**Figure 6.6: Response Time Versus Throughput**



**Figure 6.7: New Order Think Time Distribution**



**Figure 6.8: Throughput Versus Time**



Throughput versus time is graphed at 30-second intervals. Histogram bins: for Response Times .01 seconds; for Think Times .1 seconds.

---

## 6.5 Steady State Determination

*The method used to determine that the SUT had reached a steady state prior to commencing the measurement interval must be disclosed.*

The transaction throughput rate (tpmC<sub>®</sub>) and response time were relatively constant after the initial 'ramp up' period. The throughput and response time behavior were determined by examining data reported for each interval over the duration of the benchmark. Ramp up, steady state and ramp down regions are discernible in the graph, Figure 6.17.

## 6.6 Work Performed During Steady State

*A description of how the work normally performed during a sustained test (for example checkpointing, writing redo/undo log records, etc.), actually occurred during the measurement interval must be reported.*

### 6.6.2 Checkpoint Conditions

A Sybase Adaptive Server Enterprise 11.5 checkpoint forces all “dirty” pages (pages that have been updated since they were last written) to be written to the durable database devices. Checkpoints are marked by a special record written into the logs at the completion of the foregoing process.

#### 6.6.1 Checkpoint

Sybase Adaptive Server Enterprise 11.5 executes a checkpoint for the following conditions:

1. Automatically, at an interval calculated by Sybase Adaptive Server Enterprise 11.5 on the basis of system activity and the recovery interval value in the system table *syscurconfigs*. The recovery interval determines checkpoint frequency by specifying the amount of time it should take the system to recover.
2. Upon an explicit **checkpoint** request in Transact-SQL.

### 6.6.3 Checkpoint Implementation

For each benchmark measurement after all users are active, the script **checkpoints** issues a checkpoint and starts a background process, which sleeps and performs another checkpoint every 30 minutes. The recovery interval is configured large enough that no other checkpoints occur during the measurement.

## 6.7 Reproducibility

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

A second measurement achieved a throughput of 51,990.77 tpmC<sub>®</sub> during a 30-minute, steady state interval.

## 6.8 Measurement Period Duration

*A statement of the duration of the measurement interval for the reported Maximum Qualified Throughput (tpmC<sub>®</sub>) must be included.*

The measurement interval was 30 minutes.

## 6.9 Regulation of Transaction Mix

*The method of regulation of the transaction mix (e.g., card decks or weighted random distribution) must be described. If weighted distribution is used and the RTE adjusts the weights associated with each transaction type, the maximum adjustments to the weight from the initial value must be disclosed.*

---

The weighted selection method of *Clause 5.2.4.1* was used. The weights were not adjusted during the run.

## 6.10 Transaction Mix

*The percentage of the total mix for each transaction type must be disclosed.*

New-Order	44.84%
Payment	43.02%
Order-Status	4.04%
Delivery	4.05%
Stock-Level	4.05%

**Table 6.5: Transaction Mix**

## 6.11 Transaction Statistics

*The percentage of New-Order transactions rolled back as a result of invalid item number must be disclosed. The average number of order-lines entered per New-Order transaction must be disclosed. The percentage of remote order-lines entered per New-Order transaction must be disclosed. The percentage of remote Payment transactions must be disclosed. The percentage of customer selections by customer last name in the Payment and Order-Status transactions must be disclosed. The percentage of Delivery transactions skipped due to there being fewer than necessary orders in the New-Order table must be disclosed.*

See Table 3.1.

---

## 6.12 Checkpoint Count and Location

*The number of checkpoints in the measurement interval, the time in seconds from the start of the measurement interval to the first checkpoint, and the Checkpoint Interval must be disclosed.*

Times in Table 6.6 below are relative to the beginning of the driver-timed phase of the test. The checkpoint interval is 30 minutes. The first checkpoint within the 30-minute measurement interval was 595 seconds from its start. The checkpoint interval during steady-state is 30 minutes. In accord with 5.5.2.2, there is no checkpoint within the guard zones  $1800/4=450$  seconds from the beginning and end of the measurement interval.

Event	From: (Sec.)	To: (Sec.)	Duration (Sec.)
checkpoint	595	1362	767
measured interval	1815	3615	1800
checkpoint	2395	3159	764

**Table 6.6: Checkpoints**





## 7 Clause 6 Related Items

---

### 7.1 RTE Description

*If the RTE is commercially available, then its inputs must be specified. Otherwise, a description must be supplied of what inputs (e.g., scripts) to the RTE had been used. The RTE input parameters, code fragments, functions, et cetera used to generate each transaction input field must be disclosed. Comment: The intent is to demonstrate the RTE was configured to generate transaction input data as specified in Clause 2.*

The RTE (Remote Terminal Emulator) on the driver system was developed at Hewlett-Packard and is not commercially available. Appendix D lists RTE input parameters and code fragments used to generate each transaction input field.

For this instance of the TPC-C benchmark, six drivers and twelve clients were used. The drivers emulated 43,000 users logged in to the clients. An overview of the benchmark software on the drivers, clients and server is shown in Figure 7.1.

The benchmark is started with the **run** command on the driver system. **Run** controls the overall execution of the benchmark. After reading a configuration file, **run** starts TUXEDO on the client, collects pre-benchmark audit information and inserts a timestamp into a database audit table. When all the initial steps are completed, **run** invokes another program, **driver**, to start the benchmark. As the benchmark completes, **run** shuts down TUXEDO and collects the benchmark results into a single location.

---

**Driver** is the heart of the benchmark software. It simulates users as they log in, execute transactions and view results. **Driver** collects response times for each transaction and saves them in a file for future analysis.

**Qualify** is the post-processing analysis program. It produces the numerical summaries and histograms needed for the disclosure report.

## 7.2 Emulated Components

*It must be demonstrated that the functionality and performance of the components being emulated in the Driver System are equivalent to the priced system.*

In the priced configuration, workstations are connected to the clients via LANs. On the tested system, 24 LAN segments carried all the traffic between the 43,000 simulated users in the RTE system and the 12 client systems. In the priced configuration, this traffic has been divided among 4 separate LAN segments for each of the 12 clients, for a total of 48 LAN segments.

## 7.3 Functional Diagrams

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

Figures 1.1 and 1.2 (in Chapter 1) show functional diagrams of the benchmark and configured systems. A description of the RTE and benchmark software is provided above.

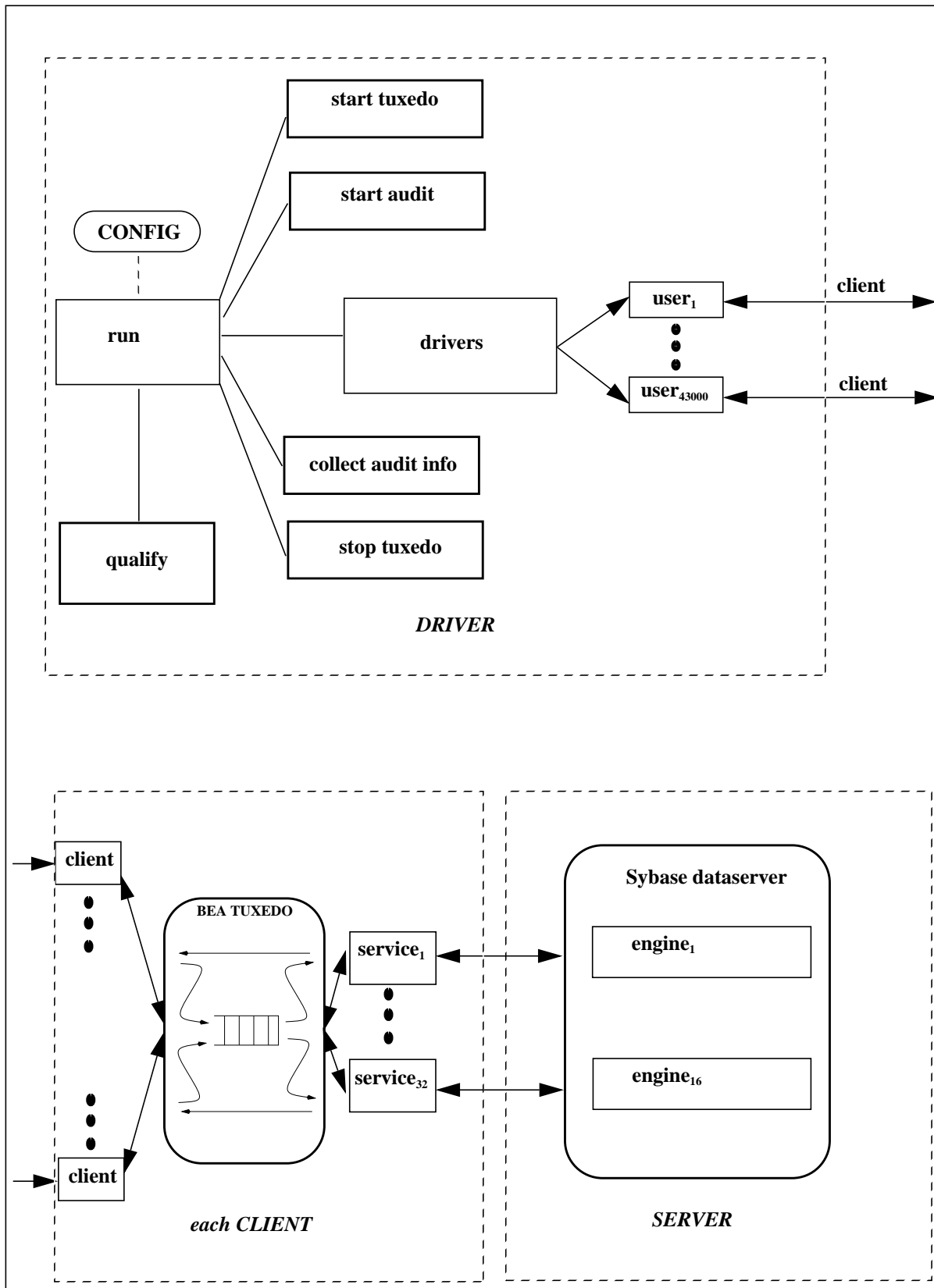
## 7.4 Networks

*The network configuration of both the tested and proposed services which are being represented and a thorough explanation of exactly which parts are being replaced with the Driver System must be disclosed.*

Figures 1.1 and 1.2 (in Chapter 1) diagram the network configurations of the benchmark and configured systems, and represent the Driver connected via LAN replacing the workstations and HUBs connected via LANs. The clients are connected via 10 BaseT to an ethernet/FDDI switch which in turn is connected via FDDI to the SUT.

*The bandwidth of the networks used in the tested/priced configurations must be disclosed.*

10BaseT local area networks (LAN) with a bandwidth of 10 megabits per second are used in the tested/priced configurations. The FDDI network used has a bandwidth of 100 megabits per second.



**Figure 7.1: Benchmark Software**



## 8 Clause 7 Related Items

---

### 8.1 System Pricing

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

*The total 5-year price of the entire configuration must be reported including: hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.*

Each priced configuration consists of an integrated system package, additional options, and components. A one (1) year warranty is standard with all Hewlett-Packard products. The hardware, software, and support/maintenance products priced in this benchmark, and pricing sources, are detailed on page vi.

### 8.2 Support Pricing

The five year support pricing for Hewlett-Packard products is based on forty-eight (48) months of monthly support costs; sixty (60) months minus the twelve month warranty period. The SYBASE support pricing is based on sixty (60) months of monthly support costs. The following support products were priced in the benchmark:

- HP four-hour onsite repair hardware support,
- HP telephone support for software and updates

- 
- Sybase Standard Technical Support and,
  - BEA TUXEDO Standard Technical Support

### **8.2.1 HP Hardware Support**

HP four-hour maximum response, onsite support for hardware provides service from 8:00 A.M. to 5:00 P.M. Monday through Friday. Service requests made as late as 5:00 P.M. will receive a response the same day.

### **8.2.2 HP Software Support**

HP Software Support provides the following:

- Access to the HP Response Centers for fault isolation and problem solving assistance,
- Guaranteed two (2) hour call return, immediate response for critical calls,
- Electronic access to product and support information,
- Electronic access to software patches,
- Right-to-use and copy software updates.

### **8.2.3 Hubs**

An additional 10% of the needed hubs were included in the priced configuration to provide the required four hour repair for hardware components. The return-for-replacement support would be used to restock spares.

### **8.2.4 Sybase Standard Technical Support**

Sybase Standard Technical Support includes:

- Product updates,
- A regular technical publication,
- Three annual training credits,
- Unlimited, toll-free telephone service to assist in product installation, syntax, and usage that is available from 7:00 A.M. to 5:30 P.M. PST Monday through Friday.

### **8.3 Discounts**

The following generally available discounts were applied to the priced configurations:

- A Sybase 10% dollar volume discount.

### **8.4 Availability**

*The committed delivery date for general availability (availability date) of products used in the price calculation 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.*

see below

### **8.5 Priced System Configuration**

The hardware, software, and support/maintenance products priced in this benchmark are detailed on page vi.

---

**8.6 Throughput,  
Price/Performance,  
and Availability Date**

*A statement of the measured tpmC<sub>®</sub> as well as the respective calculations for the 5-year pricing, price/performance (price/tpmC<sub>®</sub>).*

For Throughput and Price/Performance, please see page iv and v. The Price/Performance calculation spreadsheet appears on page vi.

All hardware components will be available March 2, 1998. OS and database software represented in this test will be available August 13, 1998.





## 9 Clause 9 Related Items

---

### 9.1 Auditor's Report

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

*If audited, the auditor's attestation letter must be made readily available to the public as part of the Full Disclosure Report, but a detailed report from the auditor is not required.*

This implementation of the TPC Benchmark C<sup>®</sup> on the HP 9000 V2250 Enterprise Server was audited by Richard Gimarc of Performance Metrics, Inc.

2229 Benicia Drive  
Suite 101  
Rancho Cordova, California 95670  
(phone) 916-635-2822

The attestation letter is shown on the following pages. A complete audit report was not generated.

This was a remote audit.

**PERFORMANCE METRICS INC.**  
**TPC Certified Auditors**

---

Len Sheppard  
 High Performance Systems Division  
 Hewlett-Packard Company  
 19111 Pruneridge Avenue  
 Cupertino, CA 95014

February 6, 1998

I have verified remotely the TPC Benchmark™ C for the following configuration:

Server: Hewlett-Packard 9000 V2250 Enterprise Server Solution  
 Database Manager: Sybase Adaptive Server Enterprise 11.5  
 Operating System: HP-UX 11.00  
 Transaction Monitor: BEA Tuxedo 4.2

CPUs	Memory	Disks	New-Order Response Time @90%	tpmC
Server: HP 9000 V2250 Enterprise Server Solution				
16 PA-RISC 8200 @ 240 MHz	Main: 16 GB I-Cache: 2 MB D-Cache: 2 MB	23 @ 4.3 GB AutoRAID 30 @ 9.1 GB AutoRAID	2.31 sec.	52,117.80
12 Clients: HP 9000 Model D230 Enterprise Server				
1 PA-RISC 7300LC @ 160 MHz	Main: 768 MB I-Cache: 64 KB D-Cache: 64 KB L2 Cache: 1 MB	1 @ 3.99 GB	n/a	n/a

In my opinion, these performance results were produced in compliance with the TPC requirements for the benchmark.

The following attributes of the benchmark were given special attention:

- The transactions were correctly implemented.
- The database files were properly sized and populated.
- The database was properly scaled with 4,300 warehouses.
- The ACID properties were met.
- Input data was generated according to the specified percentages.
- Eight hours of log space was configured on the priced system.
- The data for the 180-day space calculation was verified. To satisfy this requirement, four 4.3 GB AutoRAID devices were substituted for four 9.1 GB AutoRAID devices in the priced configuration. Performance data collected during steady state demonstrated that this disk substitution was performance neutral.

**PERFORMANCE METRICS INC.**  
**TPC Certified Auditors**

---

- Measurement cycle times did not include delays for emulated components.
- The steady state portion of the test was 30 minutes.
- One checkpoint was taken during the steady state portion of the test.
- Checkpoints were verified to be clear of the guard zones.
- There were 43,000 user contexts present on the system.
- Each emulated user had a unique starting random number seed.
- The NURand constants used for database load and run-time were verified.
- System pricing was checked for major components and maintenance.

Additional Audit Notes: none

Regards,

*Richard L. Gimarc*

Richard L. Gimarc  
Auditor



## *10 Report Availability*

---

Requests for this TPC Benchmark C® Full Disclosure Report should be sent to:

Transaction Processing Performance Council  
c/o Shanley Public Relations  
777 North First Street  
Suite 600  
San Jose, CA 95112-6311  
or your local Hewlett-Packard sales office.



# Appendix A: Client Source

---

This appendix contains the source and makefiles for the Tuxedo client and server programs. All of the programs ran on the client machine.

---

## A.1 Client Front-End

---

### *client/Makefile*

---

---

```
#####
# @(#) Version: A.10.10 $Date: 97/12/15 10:53:25 $
#
# (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#####

#
# Makefile for compiling the client, batch-tpcc, and service code
#

OH = ${ORACLE_HOME}
P = ${WORK_DIR}/src
I = $(P)/lib
L = $(P)/lib
D = $(P)/driver
Q = $(P)/que
S = $(P)/client

OPT = -Wl,-a,archive_shared +O4 +Ofastaccess +Onolimit +Oentrsched +ESlit
LD_OPTS = -ldld -a archive_shared +Oprocelim +Ofastaccess +ESlit
TUXEDO = -D_HPUX_SOURCE ${ROOTDIR}/include ${OPT}

# ORA_LOAD = -L${OH}/lib ${OH}/lib/osntab.o -lbench -locic -lsqlnet -lnetv2 -lnetwork -lora -lsqlnet -lora -lnlsrt13 -lnlsrt1 -lc3v6 -lcore3 -lcore -lm -lnlsrt3 -lnlsrt1
# -lmsg -lpls -lcore3 -lnlsrt3 -lnlsrt1 -lstublm -lc -lm

ORA_LOAD = -L${OH}/lib -lbench -lsqlnet -lclient -lserver -lcommon -lgeneric -lsqlnet -lclient -lserver -lcommon -lgeneric -lnlsrt3 -lc3v6 -lcore3 -lnlsrt3 -lcore3
/oracle/v7/lib/epcni.o -lcl -lm

LD_FLAGS_SYB= ${OPT} ${L}/tpc_lib.a -L${SYBASE}/lib -lsybdb -lm
LD_FLAGS_ORA= ${OPT} ${L}/tpc_lib.a ${ORA_LOAD}
LD_FLAGS_SQL= ${OPT} ${L}/tpc_lib.a -L/opt/odbc/lib -lodbc -lm
```

```

ORA_INCLUDE=-I${OH}/rdmbs/demo
SYB_INCLUDE=-I${SYBASE}/include
VIS_INCLUDE=-I${VISIGENIC}/include
TUX_INCLUDE=-I${ROOTDIR}/include
INCLUDE =-I${S}/oracle -I. -I$L

CFLAGS = ${OPT} ${INCLUDE} ${TUX_INCLUDE}
CFLAGS_SYB = ${OPT} ${INCLUDE} ${TUX_INCLUDE} ${SYB_INCLUDE}
CFLAGS_ORA = -Aa -D HPUX_SOURCE ${OPT} ${INCLUDE}
${ORA_INCLUDE} ${TUX_INCLUDE}
CFLAGS_SQL = -Aa -Dunix -D HPUX_SOURCE -DVG_UNIX ${OPT}
${INCLUDE} ${TUX_INCLUDE} ${SQL_INCLUDE} ${VIS_INCLUDE}

PROGRAMS = client service startup client_batch msg_server raw

tpcc_client: client
    mv client ${WORK_DIR}/bin/
others_sybase: raw startup client_batch_syb msg_server_syb
    mv raw startup client_batch_msg_server ${WORK_DIR}/bin
others_oracle: raw startup client_batch_ora msg_server_ora
    mv raw startup client_batch_msg_server ${WORK_DIR}/bin
others_sqlserver: raw startup client_batch_sql msg_server_sql
    mv raw startup client_batch_msg_server ${WORK_DIR}/bin
service_oracle: service_ora
    mv service ${WORK_DIR}/bin/
service_sybase: service_syb
    mv service ${WORK_DIR}/bin/
service_sqlserver: service_sql
    mv service ${WORK_DIR}/bin/

${S}/sybase/transaction.o: ${S}/sybase/transaction.c
    $(CC) ${CFLAGS_SYB} $(L)/tpc_lib.a -c ${S}/sybase/transaction.c;
${S}/sqlserver/transactionb.o: ${S}/sqlserver/transactionb.c
    $(CC) ${CFLAGS_SQL} $(L)/tpc_lib.a -c ${S}/sqlserver/transactionb.c;

ORA_OBJS=plnew.o plord.o plpay.o pldel.o plsto.o tpccpl.o

#transaction.o: ${S}/oracle/transaction.c
#    $(CC) ${CFLAGS_ORA} $(L)/tpc_lib.a -c ${S}/oracle/transaction.c;
plnew.o: ${S}/oracle/plnew.c
    $(CC) ${CFLAGS_ORA} $(L)/tpc_lib.a -c ${S}/oracle/plnew.c;
plord.o: ${S}/oracle/plord.c
    $(CC) ${CFLAGS_ORA} $(L)/tpc_lib.a -c ${S}/oracle/plord.c;
plpay.o: ${S}/oracle/plpay.c
    $(CC) ${CFLAGS_ORA} $(L)/tpc_lib.a -c ${S}/oracle/plpay.c;
pldel.o: ${S}/oracle/pldel.c
    $(CC) ${CFLAGS_ORA} $(L)/tpc_lib.a -c ${S}/oracle/pldel.c;
plsto.o: ${S}/oracle/plsto.c
    $(CC) ${CFLAGS_ORA} $(L)/tpc_lib.a -c ${S}/oracle/plsto.c;
tpccpl.o: ${S}/oracle/tpccpl.c
    $(CC) ${CFLAGS_ORA} $(L)/tpc_lib.a -c ${S}/oracle/tpccpl.c;

raw: raw.o
    cc ${CFLAGS} raw.o $(L)/tpc_lib.a -o raw

startup: startup.o $(L)/tpc_lib.a
    cc ${CFLAGS} startup.o $(L)/tpc_lib.a -o startup
    chmod a+rw startup

client: client.o tux_transaction.o $(L)/tpc_lib.a
    ${ROOTDIR}/bin/buildclient -v -f \
    "client.o tux_transaction.o $(L)/tpc_lib.a -lm" -o client

service_syb: service.o ${S}/sybase/transaction.o $(L)/tpc_lib.a
    ${ROOTDIR}/bin/buildserver -v -b shm \
    -s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRV_SVC \
    -o service \
    -f "service.o transaction.o $(L)/tpc_lib.a \
    ${SYBASE}/lib/libsybdb.a -lm";

service_ora: service.o transaction.o $(ORA_OBJS) $(L)/tpc_lib.a
    ${ROOTDIR}/bin/buildserver -v -b shm \
    -s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRV_SVC \
    -o service \
    -f "service.o transaction.o $(ORA_OBJS) $(L)/tpc_lib.a \
    ${LD_FLAGS_ORA}

```

```

service_sql: service.o ${S}/sqlserver/transactionb.o $(L)/tpc_lib.a
    ${ROOTDIR}/bin/buildserver -v -b shm \
    -s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRV_SVC \
    -o service \
    -f "service.o transactionb.o $(L)/tpc_lib.a \
    /vsbuild/v1.10/build/com/obj/inst/libodbc.sl"

client_batch_ora: $(D)/driver.o $(D)/generate.o transaction.o $(ORA_OBJS)
$(Q)/dummy_que.o $(L)/tpc_lib.a \
    $(L)/server_default.o
    $(CC) $(D)/driver.o $(D)/generate.o transaction.o $(ORA_OBJS)
$(Q)/dummy_que.o $(L)/server_default.o $(L)/tpc_lib.a ${LD_FLAGS_ORA} -o
client_batch;

client_batch_syb: $(D)/driver.o $(D)/generate.o transaction.o $(Q)/dummy_que.o
$(L)/tpc_lib.a \
    $(L)/server_default.o
    $(CC) $(D)/driver.o $(D)/generate.o transaction.o $(Q)/dummy_que.o
$(L)/server_default.o $(L)/tpc_lib.a ${LD_FLAGS_SYB} -o client_batch;

client_batch_sql: $(D)/driver.o $(D)/generate.o transactionb.o $(Q)/dummy_que.o
$(L)/tpc_lib.a \
    $(L)/server_default.o
    $(CC) $(D)/driver.o $(D)/generate.o transactionb.o
$(Q)/dummy_que.o $(L)/server_default.o $(L)/tpc_lib.a ${LD_FLAGS_SQL} -o
client_batch;

msg_server_ora: $(Q)/msg_server.o transaction.o $(ORA_OBJS) $(L)/tpc_lib.a
    $(CC) $(Q)/msg_server.o transaction.o $(ORA_OBJS) ${LD_FLAGS_ORA} -o
msg_server;
msg_server_syb: $(Q)/msg_server.o transaction.o $(L)/tpc_lib.a
    $(CC) $(Q)/msg_server.o transaction.o ${LD_FLAGS_SYB} -o msg_server;

msg_server_sql: $(Q)/msg_server.o transactionb.o $(L)/tpc_lib.a
    $(CC) $(Q)/msg_server.o transactionb.o ${LD_FLAGS_SQL} -o msg_server;

clean:
    rm -f *.o

cllobber: clean
    rm -f ${PROGRAMS}

```

---

## client/make\_pbo

---

```

#!/usr/bin/csh

setenv CCOPTS "+DA1.1 +DS1.1c +P +df pbo_data/client.flow.data +pgm client"
make -f Makefile tpcc_client

if (${DATABASE} == "sybase") then
    setenv CCOPTS "+DA1.1 +DS1.1c +P +df pbo_data/syb_service.flow.data +pgm
service"
    make -f Makefile service_sybase
    setenv CCOPTS "+DA1.1 +DS1.1c"
    make -f Makefile others_sybase
else if (${DATABASE} == "oracle") then
    setenv CCOPTS "+DA1.1 +DS1.1c +P +df pbo_data/ora_service.flow.data +pgm
service"
    make -f Makefile service_oracle
    setenv CCOPTS "+DA1.1 +DS1.1c"
    make -f Makefile others_oracle
else if (${DATABASE} == "sqlserver") then
    setenv CCOPTS "+DA1.1 +DS1.1c +P +df pbo_data/sql_service.flow.data +pgm
service"
    make -f Makefile service_sqlserver
    setenv CCOPTS "+DA1.1 +DS1.1c"
    make -f Makefile others_sqlserver
endif

```



---

---

## client/make\_inst

---

---

```
#!/usr/bin/csh

setenv CCOPTS "+DA1.1 +DS1.1c +I"
make -f Makefile tpcc_client

if (${DATABASE} == "sybase") then
  setenv CCOPTS "+DA1.1 +DS1.1c +I"
  make -f Makefile service_sybase
  setenv CCOPTS "+DA1.1 +DS1.1c"
  make -f Makefile others_sybase
else if (${DATABASE} == "oracle") then
  setenv CCOPTS "+DA1.1 +DS1.1c +I"
  make -f Makefile service_oracle
  setenv CCOPTS "+DA1.1 +DS1.1c"
  make -f Makefile others_oracle
else if (${DATABASE} == "sqlserver") then
  setenv CCOPTS "+DA1.1 +DS1.1c +I"
  make -f Makefile service_sqlserver
  setenv CCOPTS "+DA1.1 +DS1.1c"
  make -f Makefile others_sqlserver
endif
```

---

---

## client/client.c

---

---

```
/*
*****
@(#) Version: A.10.10 $Date: 97/12/15 10:53:26 $

(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****
History
941101 JVM Fixed login screen to detect broken connection (used to loop)
941013 JVM Added audit strings to the login form
941013 VM modified the getfield procedure to add digit and char check
according to the field type.
941014 VM added the status_msg routine to display transaction results.
941015 VM added zip routine to format zip codes and phone routine
to format phone numbers.
*****/

#include "iobuf.h"
#include "tpcc.h"
#include <signal.h>

#define until(c) while(!(c))

/* a generic transaction variable. */
generic_trans generic_transaction;
generic_trans *trans=&generic_transaction;

/* global variables set up during initialization */
int user;
ID warehouse;
ID district;

main(argc, argv)
int argc;
char **argv;
```

```
{
int key;

/* setup the transactions */
key = setup(argc, argv);

/* repeat until done */
while (key != '9' && key != EOF)
{

/* get the menu choice */
key = menu_read();

/* process according to the choice */
switch(key)
{
case '1': key = neworder(&trans->neworder); break;
case '2': key = payment(&trans->payment); break;
case '3': key = ordstat(&trans->ordstat); break;
case '4': key = delivery(&trans->delivery); break;
case '5': key = stocklev(&trans->stocklev); break;
case EOF: break;
case '9': break;
default: msgline("Please enter a valid menu choice");
}
}

/* done */
cleanup();
}
```

```
/*
*****
*****
**
```

### Neworder form processing

```
*****
**
*****
**/
```

```
define_iobuf(neworder_form, 900);
```

```
int neworder(trans)
neworder_trans *trans;
{
int key;
display(neworder_form);
key = neworder_read(trans);
if (key != ENTER) return key;
neworder_transaction(trans);
neworder_write(trans);
return key;
}
```

```
int neworder_read(trans)
neworder_trans *trans;
{
int i;
int field;
int key;
int ol;
```

```
/* Our warehouse number is fixed */
trans->W_ID = warehouse;
trans->D_ID = EMPTY_NUM;
```

```
/* assume nothing set yet */
trans->C_ID = EMPTY_NUM;
for (i=0; i<15; i++)
{
trans->item[i].OL_I_ID = EMPTY_NUM;
trans->item[i].OL_QUANTITY = EMPTY_NUM;
trans->item[i].OL_SUPPLY_W_ID = EMPTY_NUM;
}
```

```

/* Process fields until done */
for (field = 1; field > 0; field = next_field(field, key, 47))
  retry: switch (field)
  {
    case 1: key = read_number(4, 29, &trans->D_ID, 2);
            break;

    case 2: key = read_number(5, 12, &trans->C_ID, 4);
            break;

    case 3: case 6: case 9: case 12: case 15:
    case 18: case 21: case 24: case 27: case 30:
    case 33: case 36: case 39: case 42: case 45:
      ol = (field - 3) / 3;
      key = read_number(9+ol, 3, &trans->item[ol].OL_SUPPLY_W_ID,4);
      break;

    case 4: case 7: case 10: case 13: case 16:
    case 19: case 22: case 25: case 28: case 31:
    case 34: case 37: case 40: case 43: case 46:
      ol = (field - 3) / 3;
      key = read_number(9+ol,10, &trans->item[ol].OL_I_ID, 6);
      break;

    case 5: case 8: case 11: case 14: case 17:
    case 20: case 23: case 26: case 29: case 32:
    case 35: case 38: case 41: case 44: case 47:
      ol = (field - 3) / 3;
      key = read_number(9+ol, 45, &trans->item[ol].OL_QUANTITY, 2);
      break;
  }

/* abort the screen if requested */
if (key != ENTER)
  return key;

/* calculate how many items were entered */
for (i=15; i>0; i--)
  if ((trans->item[i-1].OL_I_ID != EMPTY_NUM) ||
      (trans->item[i-1].OL_SUPPLY_W_ID != EMPTY_NUM) ||
      (trans->item[i-1].OL_QUANTITY != EMPTY_NUM)) break;
trans->O_OL_CNT = i;

/* make sure all necessary fields are filled in */
if (trans->D_ID == EMPTY_NUM)
  {field=1; msgline("Please specify district"); goto retry;}
if (trans->C_ID == EMPTY_NUM)
  {field=2; msgline("Please specify customer id"); goto retry;}
if (trans->O_OL_CNT == 0)
  {field=3; msgline("Please enter at least one orderline"); goto retry;}
for (i=0; i<trans->O_OL_CNT; i++)
  {
    if (trans->item[i].OL_SUPPLY_W_ID == EMPTY_NUM)
      {field=i*3+3; msgline("Please enter supply warehouse"); goto retry;}
    if (trans->item[i].OL_I_ID == EMPTY_NUM)
      {field=i*3+4; msgline("Please enter Item id"); goto retry;}
    if (trans->item[i].OL_QUANTITY == EMPTY_NUM
        || trans->item[i].OL_QUANTITY <= 0)
      {field=i*3+5; msgline("Please enter quantity > 0"); goto retry;}
  }

/* decide if they were all local */
for (i=0; i<trans->O_OL_CNT; i++)
  if (trans->item[i].OL_SUPPLY_W_ID != trans->W_ID) break;
trans->all_local = (i == trans->O_OL_CNT);

/* display number of order lines */
number(6, 42, trans->O_OL_CNT, 2);

msgline("");
flush();
return key;
}

```

```
neworder_write(t)
```

```

neworder_trans *t;
{
  int i;
  MONEY amount, total_amount, cost;

  /* Rev. 3.3 error checking: both of the following branches are
  * skipped. We'll go to status and print an error message.
  */

  /* CASE: invalid item, display only these values */
  if (t->status == E_INVALID_ITEM)
  {
    text(5, 25, t->C_LAST);
    text(5,52, t->C_CREDIT);
    number(6, 15, t->O_ID, 8);
  }

  /* CASE: everything OK, display everything */
  else if (t->status == OK)
  {
    text(5, 25, t->C_LAST);
    text(5,52, t->C_CREDIT);
    number(6, 15, t->O_ID, 8);
    date(4, 61, t->O_ENTRY_D);
    real(5, 64, t->C_DISCOUNT * 100, 5, 2);
    real(6, 59, t->W_TAX*100, 5, 2);
    real(6, 74, t->D_TAX*100, 5, 2);

    total_amount = 0;
    for (i=0; i < t->O_OL_CNT; i++)
    {
      /* keep track of amount of each line and total */
      amount = t->item[i].I_PRICE * t->item[i].OL_QUANTITY;
      total_amount += amount;

      /* display the item line */
      text(9+i, 19, t->item[i].I_NAME);
      number(9+i, 51, t->item[i].S_QUANTITY, 3);
      position(9+i, 58); pushc(t->item[i].brand_generic);
      money(9+i, 62, t->item[i].I_PRICE, 7);
      money(9+i, 71, amount, 8);
    }

    /* Clear the screen of any empty input fields */
    clear_screen();

    /* display the total cost */
    text(24, 63, "Total:");
    cost = total_amount * (1 - t->C_DISCOUNT) * (1 + t->W_TAX + t->D_TAX);
    money(24, 71, cost, 9);
  }

  /* display the status message */
  status(24, 1, t->status);
}

neworder_setup()
{
  int item;
  iobuf *old;

  /* start with an empty form */
  reset(neworder_form);

  /* redirect the data to a special menu buffer */
  old = out_buf; out_buf = neworder_form;

  /* clear the iobuf below the menu */
  position(3,1);
  clear_screen();

  /* set up all the field labels */
  text(3, 36, "New Order");
  text(4, 1, "Warehouse");
  number(4, 12, warehouse, 4);
  text(4, 19, "District");
  empty(4, 29, 2);
}

```

```

text(4, 55, "Date:");
text(5, 1, "Customer:");
empty(5, 12, 4);
text(5, 19, "Name:");
text(5, 44, "Credit:");
text(5, 57, "Disc:");
text(6, 1, "Order Number:");
text(6, 25, "Number of Lines:");
text(6, 52, "W_Tax:");
text(6, 67, "D_Tax:");
text(8, 2, "Supp_W Item_Num Item_Name");
text(8, 45, "Qty Stock B/G Price Amount");

/* display blank fields for each item */
for (item = 1; item <= 15; item++)
{
    empty(8+item, 3, 4);
    empty(8+item, 10, 6);
    empty(8+item, 45, 2);
}

trigger();

/* restore to the previous I/O buffer */
out_buf = old;
}

/*****
**
**
**
**
**
**
*****/

Payment form processing

/*****
**
**
**
**
*****/

define_iobuf(payment_form, 400);

int payment(trans)
payment_trans *trans;
{
    int key;
    display(payment_form);
    key = payment_read(trans);
    if (key != ENTER) return key;
    payment_transaction(trans);
    payment_write(trans);
    return key;
}

payment_setup()
{
    int item;
    iobuf *old;

    /* start with an empty form */
    reset(payment_form);

    /* redirect the data to a special menu buffer */
    old = out_buf; out_buf = payment_form;

    /* clear the iobuf below the menu */
    position(3,1);
    clear_screen();

    /* set up all the field labels */
    text(3, 38, "Payment");
    text(4, 1, "Date:");
    text(6, 1, "Warehouse:");
    number(6, 12, warehouse, 4);
    text(6, 42, "District:");

    empty(6, 52, 2);
    text(11, 1, "Customer:");
    empty(11, 11, 4);
    text(11, 17, "Cust-Warehouse:");
    empty(11, 33, 4);
    text(11, 39, "Cust-District:");
    empty(11, 54, 2);
    text(12, 1, "Name:");
    empty(12, 29, 16);
    text(12, 50, "Since:");
    text(13, 50, "Credit:");
    text(14, 50, "%Disc:");
    text(15, 50, "Phone:");
    text(17, 1, "Amount Paid:");
    empty(17, 23, 8);
    text(17, 37, "New Cust-Balance:");
    text(18, 1, "Credit Limit:");
    text(20, 1, "Cust-Data:");
    trigger();

    out_buf = old;
}

int payment_read(t)
payment_trans *t;
{
    int i;
    int field;
    int key;

    /* Our warehouse number is fixed */
    t->W_ID = warehouse;
    t->C_ID = EMPTY_NUM;
    t->D_ID = EMPTY_NUM;
    t->C_W_ID = EMPTY_NUM;
    t->C_D_ID = EMPTY_NUM;
    t->H_AMOUNT = EMPTY_FLT;
    t->C_LAST[0] = '\0';

    /* Process fields until done */
    for (field = 1; field > 0; field = next_field(field, key, 6))
        retry: switch (field)
        {
            case 1: key = read_number(6, 52, &t->D_ID, 2);
                break;

            case 2:
                /* if last name specified, skip this field */
                if (t->C_LAST[0] != '\0')
                    break;

                /* read in the customer id */
                key = read_number(11, 11, &t->C_ID, 4);

                /* if specified, don't allow last name to be entered */
                if (t->C_ID != EMPTY_NUM)
                {
                    blanks(12, 29, 16);
                    t->C_LAST[0] = '\0';
                }

                /* refresh the C_LAST underlines, if possibly needed */
                else if (t->C_LAST[0] == '\0')
                    empty(12, 29, 16);
                break;

            case 3: key = read_number(11, 33, &t->C_W_ID, 4);
                break;

            case 4: key = read_number(11, 54, &t->C_D_ID, 2);
                break;

            case 5:
                /* skip this field if C_ID was already specified */
                if (t->C_ID != EMPTY_NUM)
                    break;

```

```

/* read in the customer last name */
key = read_text(12, 29, t->C_LAST, 16);

/* if specified, don't allow c_id to be entered */
if (t->C_LAST[0] != '\0')
{
    blanks(11, 11, 4);
    t->C_ID = EMPTY_NUM;
}

/* refresh the C_ID underlines, if possibly needed */
else if (t->C_ID == EMPTY_NUM)
    empty(11, 11, 4);
break;

case 6: key = read_money(17, 23, &t->H_AMOUNT, 8);
    break;
}

/* if Aborted, then done */
if (key != ENTER)
    return key;

/* Make sure all the fields were entered */
if (t->D_ID == EMPTY_NUM)
    {field=1; msgline("Please enter district id"); goto retry;}
if (t->C_ID == EMPTY_NUM && t->C_LAST[0] == '\0')
    {field=2; msgline("C_ID or C_LAST must be entered"); goto retry;}
if (t->C_W_ID == EMPTY_NUM)
    {field=3; msgline("Please enter customer's warehouse"); goto retry;}
if (t->C_D_ID == EMPTY_NUM)
    {field=4; msgline("please enter customer's district"); goto retry;}
if (t->H_AMOUNT == EMPTY_FLT)
    {field=6; msgline("Please enter payment amount"); goto retry;}
if (t->H_AMOUNT <= 0)
    {field=6; msgline("Please enter a positive payment"); goto retry;}

t->byname = (t->C_ID == EMPTY_NUM);
msgline("");
flush();
return key;
}

payment_write(t)
payment_trans *t;
{
    /* if errors, display a message and quit */
    if (t->status != OK)
    {
        status(24, 1, t->status);
        return;
    }

    /* display the screen */
    date(4, 7, t->H_DATE);
    text(7, 1, t->W_STREET_1);
    text(7, 42, t->D_STREET_1);
    text(8, 1, t->W_STREET_2);
    text(8, 42, t->D_STREET_2);
    text(9, 1, t->W_CITY);
    text(9, 22, t->W_STATE);
    zip(9, 25, t->W_ZIP);
    text(9, 42, t->D_CITY);
    text(9, 63, t->D_STATE);
    zip(9, 66, t->D_ZIP);
    number(11, 11, t->C_ID, 4);
    text(12, 9, t->C_FIRST);
    text(12, 26, t->C_MIDDLE);
    text(12, 29, t->C_LAST);
    date_only(12, 58, t->C_SINCE);
    text(13, 9, t->C_STREET_1);
    text(13, 58, t->C_CREDIT);
    text(14, 9, t->C_STREET_2);
    real(14, 58, t->C_DISCOUNT*100, 5, 2); /* percentage or fraction? */

```

```

text(15, 9, t->C_CITY);
text(15, 30, t->C_STATE);
zip(15, 33, t->C_ZIP);
phone(15, 58, t->C_PHONE);
money(17, 17, t->H_AMOUNT, 14);
money(17, 55, t->C_BALANCE, 15);
money(18, 17, t->C_CREDIT_LIM, 14);

/* Display cust data if bad credit. */
if (t->C_CREDIT[0] == 'B' && t->C_CREDIT[1] == 'C')
    long_text(20, 12, t->C_DATA, 50);
}

/*****
*****
*****
*****

ORDSTAT form processing

*****
*****
*****

define_iobuf(ordstat_form, 300);

int ordstat(t)
ordstat_trans *t;
{
    int key;
    display(ordstat_form);
    key = ordstat_read(trans);
    if (key != ENTER) return key;
    ordstat_transaction(trans);
    ordstat_write(trans);
    return key;
}

ordstat_setup()
{
    int item;
    iobuf *old;

    /* start with an empty form */
    reset(ordstat_form);

    /* redirect the data to a special menu buffer */
    old = out_buf; out_buf = ordstat_form;

    /* clear the iobuf below the menu */
    position(3,1);
    clear_screen();

    /* set up all the field labels */
    text(3, 35, "Order-Status");
    text(4, 1, "Warehouse:");
    number(4, 12, warehouse, 4);
    text(4, 19, "District:");
    empty(4, 29, 2);
    text(5, 1, "Customer:");
    empty(5, 11, 4);
    text(5, 18, "Name:");
    empty(5, 44, 16);
    text(6, 1, "Cust-Balance:");
    text(8, 1, "Order-Number");
    text(8, 26, "Entry-Date:");
    text(8, 60, "Carrier-Number:");
    text(9, 1, "Supply-W");
    text(9, 14, "Item-Num");
    text(9, 25, "Qty");
    text(9, 33, "Amount");
    text(9, 45, "Delivery-Date");

    trigger();

```

```

/* done */
out_buf = old;
}

int ordstat_read(t)
ordstat_trans *t;
{
int i;
int field;
int key;

/* Our warehouse number is fixed */
t->W_ID = warehouse;
t->C_ID = EMPTY_NUM;
t->D_ID = EMPTY_NUM;
t->C_LAST[0] = '\0';

/* Process fields until done */
for (field = 1; field > 0; field = next_field(field, key, 3))
retry: switch (field)
{

case 1: key = read_number(4, 29, &t->D_ID, 2);
break;

case 2:
/* if last name specified, skip this field */
if (t->C_LAST[0] != '\0')
break;

/* read in the customer id */
key = read_number(5, 11, &t->C_ID, 4);

/* if specified, don't allow last name to be entered */
if (t->C_ID != EMPTY_NUM)
{
blanks(5, 44, 16);
t->C_LAST[0] = '\0';
}

/* refresh the C_LAST underlines, if possibly needed */
else if (t->C_LAST[0] == '\0')
empty(5, 44, 16);
break;

case 3:
/* skip this field if C_ID was already specified */
if (t->C_ID != EMPTY_NUM)
break;

/* read in the customer last name */
key = read_text(5, 44, t->C_LAST, 16);

/* if specified, don't allow c_id to be entered */
if (t->C_LAST[0] != '\0')
{
blanks(5, 11, 4);
t->C_ID = EMPTY_NUM;
}

/* refresh the C_ID underlines, if possibly needed */
else if (t->C_ID == EMPTY_NUM)
empty(5, 11, 4);
break;
}

/* if Aborted, then done */
if (key != ENTER)
return key;

/* ensure all the necessary fields were entered */
if (t->D_ID == EMPTY_NUM)
{field=1; msgline("Please enter district id"); goto retry;}
if (t->C_ID == EMPTY_NUM && t->C_LAST[0] == '\0')
{field=2; msgline("C_ID or C_LAST must be entered"); goto retry;}

```

```

t->byname = (t->C_ID == EMPTY_NUM);
msgline("");
flush();
return key;
}

```

```

ordstat_write(t)
ordstat_trans *t;
{
int i;

/* if errors, display a status message and quit */
if (t->status != OK)
{
status(24, 1, t->status);
return;
}

```

```

/* display the results */
number(5, 11, t->C_ID, 4);
text(5, 24, t->C_FIRST);
text(5, 41, t->C_MIDDLE);
text(5, 44, t->C_LAST);
money(6, 15, t->C_BALANCE, 10);
number(8, 15, t->O_ID, 8);
date(8, 38, t->O_ENTRY_DATE);
if (t->O_CARRIER_ID > 0)
number(8, 76, t->O_CARRIER_ID, 2);

for (i=0; i < t->ol_cnt; i++)
{
number(i+10, 3, t->item[i].OL_SUPPLY_W_ID, 4);
number(i+10, 14, t->item[i].OL_I_ID, 6);
number(i+10, 25, t->item[i].OL_QUANTITY, 2);
money(i+10, 32, t->item[i].OL_AMOUNT, 9);
date_only(i+10, 47, t->item[i].OL_DELIVERY_DATE);
}

```

```

*****
*****

```

delivery form processing

```

*****
*****

```

```

define_iobuf(delivery_form, 300);

```

```

int delivery(t)
delivery_trans *t;
{
int key;
display(delivery_form);
key = delivery_read(trans);
if (key != ENTER) return key;
delivery_enqueue(trans);
delivery_write(trans);
return key;
}

```

```

delivery_setup()
{
int item;
iobuf *old;

```

```

/* start with an empty form */
reset(delivery_form);

```

```

/* redirect the data to a special menu buffer */
old = out_buf; out_buf = delivery_form;

/* clear the iobuf below the menu */
position(3,1);
clear_screen();

/* set up all the field labels */
text(3, 38, "Delivery");
text(4, 1, "Warehouse:");
number(4, 12, warehouse, 4);
text(6, 1, "Carrier Number:");
empty(6, 17, 2);

trigger();

/* done */
out_buf = old;
}

int delivery_read(t)
delivery_trans *t;
{
int i;
int field;
int key;

/* Our warehouse number is fixed */
t->W_ID = warehouse;
t->O_CARRIER_ID = EMPTY_NUM;

/* Process fields until done */
for (field = 1; field > 0; field = next_field(field, key, 1))
retry: switch (field)
{
case 1: key = read_number(6, 17, &t->O_CARRIER_ID, 2);
break;
}

/* if Aborted, then done */
if (key != ENTER)
return key;

/* Must enter the carrier id */
if (t->O_CARRIER_ID == EMPTY_NUM)
{field=1; msgline("Please enter the Carrier Number"); goto retry; }

/* clear the message line */
msgline("");
flush();
return key;
}

delivery_write(t)
delivery_trans *t;
{
if (t->status == OK)
text(8, 1, "Execution Status: Delivery has been queued");
else
status(8, 1, t->status);
}

*****
*****

stocklev form processing

*****
*****

define_iobuf(stocklev_form, 300);

int stocklev(t)
stocklev_trans *t;
{
int key;
display(stocklev_form);
key = stocklev_read(trans);
if (key != ENTER) return key;
stocklev_transaction(trans);
stocklev_write(trans);
return key;
}

stocklev_setup()
{
int item;
iobuf *old;

/* start with an empty form */
reset(stocklev_form);

/* redirect the data to a special menu buffer */
old = out_buf; out_buf = stocklev_form;

/* clear the iobuf below the menu */
position(3,1);
clear_screen();

/* set up all the field labels */
text(3, 35, "Stock-Level");
text(4, 1, "Warehouse:");
number(4, 12, warehouse, 4);
text(4, 19, "District:");
number(4, 29, district, 2);
text(6, 1, "Stock Level Threshold:");
empty(6, 24, 2);
text(8, 1, "low stock");

trigger();

/* done */
out_buf = old;
}

int stocklev_read(t)
stocklev_trans *t;
{
int field;
int key;

t->W_ID = warehouse;
t->D_ID = district;
t->threshold = EMPTY_NUM;

/* Process fields until done */
for (field = 1; field > 0; field = next_field(field, key, 1))
retry: switch (field)
{
case 1: key = read_number(6, 24, &t->threshold, 2);
break;
}

/* if Aborted, then done */
if (key != ENTER)
return key;

/* make sure the necessary fields were entered */
if (t->threshold == EMPTY_NUM || t->threshold <= 0)
{field=1; msgline("Please enter a threshold > 0"); goto retry; }

/* clear the message line */
msgline("");
}

```

```

flush();
return key;
}

stocklev_write(t)
stocklev_trans *t;
{
if (t->status == OK)
    number(8, 12, t->low_stock, 3);
else
    status(10, 1, t->status);
}

/*****
**
*****/

login form processing

/*****
**
*****/

int login()
{
    int field;
    int key;
    char auditstr[21];
    int w_id, d_id;

    /* assume the default values */
    w_id = warehouse;
    d_id = district;
    auditstr[0] = '\0';

    /* display the login menu */
    position(1,1); clear_screen();
    text(3,30, "Please login.");
    text(5,5, "Warehouse:");
    number(5, 16, w_id, 4);
    text(5, 24, "District:");
    number(5, 34, d_id, 2);
    text(15, 5, "Audit String:");
    text(15, 19, CLIENT_AUDIT_STRING);
    empty(16, 19, 20);
    trigger();

    /* Get values until done */
    for (field = 1; field > 0; field = next_field(field, key, 3))
        retry: switch (field)
            {
            case 1:
                key = read_number(5, 16, &w_id, 4, Num);
                break;

            case 2:
                key = read_number(5, 34, &d_id, 2, Num);
                break;

            case 3:
                key = read_text(16, 19, auditstr, 20);
                break;
            }

    if (key != ENTER)
        return EOF;

    if (w_id == EMPTY_NUM && warehouse == EMPTY_NUM)
    {
        msgline("You must enter a warehouse id");
        field = 1;
        goto retry;
    }

    if (d_id == EMPTY_NUM && district == EMPTY_NUM)
    {
        msgline("You must enter a district id");
        field = 2;
        goto retry;
    }

    if (w_id != EMPTY_NUM)
        warehouse = w_id;
    if (d_id != EMPTY_NUM)
        district = d_id;

    /* done */
    flush();
    return key;
}

/*****
**
*****/

menu form processing

/*****
**
*****/

menu_setup()
{
    /* display the menu on the iobuf -- never erased */
    position(1, 1);
    clear_screen();
    string("(1)New-Order (2)Payment (3)Order-Status ");
    string("(4)Delivery (5)StockLevel (9)Exit");
}

int menu_read()
{
    position(1, 1);
    trigger();
    return getkey();
}

int next_field(current, key, max)
int current;
int key;
int max;
{
    if (key == BACKTAB)
        if (current == 1) return max;
        else return current-1;
    else if (key == TAB)
        if (current == max) return 1;
        else return current+1;
    else
        return 0;
}

msgline(str)
char *str;
{
    position(24, 1);
    clear_screen();
    string(str);
    flush(); /* Needed? */
}

```

```

int setup(argc, argv)
int argc;
char **argv;
{
int key;

/* Ignore SIGPIPE, since they occur normally */
signal(SIGPIPE, SIG_IGN);

/* get the user, warehouse and district numbers */
warehouse = EMPTY_NUM;
district = EMPTY_NUM;
key = login();
user = warehouse*DIST_PER_WARE + district + 1;

/* set up the forms */
menu_setup();
neworder_setup();
payment_setup();
ordstat_setup();
delivery_setup();
stocklev_setup();

/* connect to the delivery queue */
delivery_init(user);

/* connect to the transaction processor */
transaction_begin(user);

return key;
}

```

```

cleanup()
{

```

```

/* detach from transaction engine */
transaction_done();

```

```

/* detach from the delivery queue */
delivery_done();

```

```

/* clear the screen */
position(1, 1);
clear_screen();
flush();
}

```

```

/*****
**
*****/

```

#### Screen Output Routines

```

*****/

```

```

number(row, col, n, width)
int row;
int col;
int n;
int width;
{
char str[81];
fmt_num(str, n, width);
text(row, col, str);
}

```

```

real(row, col, x, width, dec)

```

```

int row;
int col;
double x;
int width;
int dec;
{
char str[81];
fmtflt(str, x, width, dec);
text(row, col, str);
}

```

```

date(row, col, date_str)
int row;
int col;
char *date_str;
{
text(row, col, date_str);
}

```

```

date_only(row, col, date_str)
int row;
int col;
char *date_str;
{
date_str[10] = '\0';
text(row, col, date_str);
}

```

```

money(row, col, x, width)
int row;
int col;
double x;
int width;
{
char str[81];
fmt_money(str, x, width);
text(row, col, str);
}

```

```

long_text(row, col, str, width)
int row, col, width;
char *str;
{
int pos;

```

```

/* repeat until the entire string is written out */
for (pos = width; *str != '\0'; str++, pos++)
{

```

```

/* if at end of line, position the cursor to next line */
if (pos >= width)
{
position(row, col);
pos = 0;
row++;
}
}

```

```

/* output the next character */
pushc(*str);
}
}

```

```

text(row, col, str)
int row;
int col;
char str[];
{
position(row, col);
string(str);
}

```

```

phone(row, col, str)
int row;
int col;
char *str;

```



```

{
char temp[30];

fmt_phone(temp,str);
text(row,col,temp);
}

zip(row, col, str)
int row;
int col;
char *str;
{
char temp[30];

fmt_zip(temp,str);
text(row,col,temp);
}

empty(row, col, len)
int row;
int col;
int len;
{
position(row, col);
while (len-- > 0)
pushc('_');
}

blanks(row, col, len)
int row, col, len;
{
position(row, col);
while (len-- > 0)
pushc(' ');
}

status(row, col, status)
/******
*
status displays the transaction status
Note: must correspond to 'get_status' in driver/keystroke.c
*****
int row, col;
int status;
{
text(row, col, "Execution Status: ");

if (status == OK)
string("Transaction Committed");
else if (status == E_INVALID_ITEM)
string("Item number is not valid");
/* Do the rev. 3.3 error checking here.*/
else if (status == E_INVALID_INPUT)
string("Invalid input, transaction not executed");
else
{
string("Rollback -- ");
number(row, col+30, status, 5);
}
}

/******
****
*****
****
*/

ASCII terminal control

*****
****
*****
****
*/

trigger()

```

```

/******
**
trigger sends a turnaround sequence to let the driver know to send input
*****
**/
{
pushc(TRIGGER);
}

position(row, col)
/******
**
position positions the cursor at the given row and column
*****
*/
int row;
int col;
{
pushc(ESCAPE);
pushc('[');
if (row >= 10)
pushc('0' + row/10);
pushc('0' + row%10);
pushc(';');
if (col >= 10)
pushc('0' + col/10);
pushc('0' + col%10);
pushc('H');
}

clear_screen()
/******
*
clear_screen clears the iobuf from cursor position to end of iobuf
*****
**/
{
pushc(ESCAPE);
pushc('[');
pushc('J');
}

/******
**
Screen Input Routines
*****
**/
#define funny(key) (key != ENTER && key != TAB && key != BACKTAB)

read_number(row, col, n, width)
/******
**
read_number reads an integer field
*****
*/
int row;
int col;
int *n;
int width;
{
char temp[81];
int key;
int err;
debug("read_number: row=%d col=%d width=%d n=%d\n",row, col,width,*n);

/* generate the current characters */
fmt_num(temp, *n, width);
err = NO;

/* repeat until a valid number or a funny key is pressed */
for (;;)
{
/* Let the user edit the field */
key = getfield(row, col, temp, width, Num);
if (funny(key)) return key;
}
}

```

```

/* convert the field to a number */
*n = cvt_num(temp);
if (*n != INVALID_NUM) break;

msgline("Invalid digit entered");
pushc(BELL);
err = YES;
}

/* display the new number */
number(row, col, *n, width);
if (err) msgline("");
debug("read_number: n=%d key=%d\n", *n, key);
return key;
}

int read_money(row, col, m, width)
int row;
int col;
double *m;
int width;
{
char temp[81];
int key;
int err;

err = NO;
fmt_money(temp, *m, width);

/* repeat until a valid number or a funny key is pressed */
for (;;)
{
key = getfield(row, col, temp, width, Money);
if (funny(key)) return key;

*m = cvt_money(temp);
if (*m != INVALID_FLT) break;

msgline("Please enter amount $99999.99");
pushc(BELL);
err = YES;
}

money(row, col, *m, width);
if (err) msgline("");
return key;
}

int read_real(row, col, x, width, dec)
int row, col, width;
double *x;
{
char temp[81];
int key;
int err;

/* generate the current characters */
fmt_ft(temp, *x, width, dec);
err = NO;

/* repeat until a valid number or a funny key is pressed */
for (;;)
{
key = getfield(row, col, temp, width);
if (funny(key)) return key;

/* convert the field to a number */
*x = cvt_ft(temp);
if (*x != INVALID_FLT) break;

msgline("Please enter a valid floating pt number");
pushc(BELL);
err = YES;
}

/* display the new number */
real(row, col, *x, width, dec);
if (err) msgline("");

return key;
}

int read_text(row, col, s, width)
int row, col, width;
char *s;
{
char temp[81];
int key;
int i;

/* generate the current characters */
fmt_text(temp, s, width);

/* let the user edit the field */
key = getfield(row, col, temp, width, Text);
if (funny(key)) return key;

/* Strip off leading and trailing space characters */
cvt_text(temp, s);

/* redisplay the current text */
fmt_text(temp, s, width);
text(row, col, temp);

return key;
}

int getfield(row, col, buf, width,ftype)
int row, col, width;
char buf[];
FIELD_TYPE ftype;
{
int pos, key;

debug("getfield: width=%d buf=%s\n", width, width, buf);

/* go to the beginning of the field */
position(row, col);
pos = 0;

/* repeat until a special control character is pressed */
for (;;)
{

/* get the next character */
key = getkey();

/* CASE: Add to buf if it fits and Is it a valid character ? */
if (pos < width && valid_char(key, ftype))
{
buf[pos] = key;
pos++;
pushc(key);
}

/* CASE: char is BACKSPACE. Erase last character. */
else if (key == BACKSPACE && pos > 0)
{
pos--;
buf[pos] = '_';
pushc(BACKSPACE);
pushc('_');
pushc(BACKSPACE);
}

/* CASE: enter, tab, backtab, ^c. Exit loop */
else if (key == ENTER || key == TAB || key == BACKTAB || key == CNTRL_C
|| key == EOF)
break;
}
}

```

```

else if (key=='\031') /* for debugging, let ^X == ENTER */
    {key=ENTER; break;}

/* Otherwise, ignore the character and beep */
else
    pushc(BELL);
}

debug("getfield: final key: %d buf=%s\n", key, width, buf);
return key;
}

int valid_char(key, ftype)
/******
**
valid_char is true if the key is valid for this type of field
*****
***/
int key;
FIELD_TYPE ftype;
{
int valid;
switch(ftype)
{
case Num : valid = (isdigit(key) || key == '-' || key == '.');
break;

case Text : valid = (isprint(key) || key == ' ');
break;

case Money : valid = (isdigit(key) || key == '-' || key == '.'
|| key == '$' || key == ' ');
break;

default : valid = NO;
break;
}

return valid;
}

```

```

#ifndef DEBUG
#define debug printf
#else
#define debug (void)
#endif

#include <stdio.h>

typedef int ID; /* All id's */
typedef double MONEY; /* Large integer number of cents */
typedef char TEXT; /* Add an extra byte for null terminator */
typedef double TIME; /* Elapsed seconds from start of run (float?) */
typedef int COUNT; /* integer numbers of things */
typedef double REAL; /* real numbers */
typedef int LOGICAL; /* YES or NO */
typedef struct { /* days and seconds since Jan 1, 1900 */
int day; /* NULL represented by negative day */
int sec;
} DATE;

/* Macro to convert time of day to TIME */
#include <time.h>
extern struct timeval start_time;
#define elapsed_time(t) (((t)->tv_sec - start_time.tv_sec) + \
((t)->tv_usec - start_time.tv_usec) / 1000000.0)

typedef enum {Num,Money,Text,Time,Real,Date} FIELD_TYPE; /* screen field
types */

/* Various TPCC constants */
#define W_ID_LEN 4
#define D_ID_LEN 2
#define C_ID_LEN 4
#define I_ID_LEN 6
#define OL_QTY_LEN 2
#define PMT_LEN 7
#define C_ID_LEN 4
#define C_LAST_LEN 16
#define CARRIER_LEN 2
#define THRESHOLD_LEN 2
#define DIST_PER_WARE 10
#define CUST_PER_DIST 3000
#define ORD_PER_DIST 3000
#define MAXITEMS 100000
#define MAX_DIGITS 3 /* # of digits of the NURand number selected
to generate the customer last name */
#define MAXWAREHOUSE 2000 /* maximum # of warehouses - scaling factor */
#define LOADSEED 42 /* # of digits of the NURand number selected

```

## A.2 tpc\_lib Source

### lib/tpcc.h

```

/******
**
@(#) Version: A.10.10 $Date: 97/12/15 14:01:49 $

(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****
***/
#ifndef TPCC_INCLUDED
#define TPCC_INCLUDED
#include <values.h>

/* The auditor can define these 20 char strings to be anything */
#define DRIVER_AUDIT_STRING "driver audit string"
#define CLIENT_AUDIT_STRING "client audit string"

```

```

/******
**
database identifiers and populations
*****
***/

int no_warehouse; /* scaling factor */
int no_item; /* 100000 */
int no_dist_pw; /* 10 */
int no_cust_pd; /* 3000 */
int no_ord_pd; /* 3000 */
int no_new_pd; /* 900 */
int tpcc_load_seed; /* 900 */

/* fields to add to each transaction for acid testing */
#define ACID_STUFF \
char acid_txn[2]; \
int acid_timing; \
int acid_action; \
FILE *acid_res

typedef struct {
ID OL_SUPPLY_W_ID;
ID OL_I_ID;
TEXT I_NAME[24+1];
COUNT OL_QUANTITY;
COUNT S_QUANTITY;
MONEY I_PRICE;
char brand_generic;
} neworder_item;

```

```

typedef struct {
    int status;
    LOGICAL all_local;
    ID W_ID;
    ID D_ID;
    ID C_ID;
    TEXT C_LAST[C_LAST_LEN+1];
    TEXT C_CREDIT[2+1];
    REAL C_DISCOUNT;
    COUNT O_OL_CNT;
    ID O_ID;
    TEXT O_ENTRY_D[20]; /* dates as text fields */
    REAL W_TAX;
    REAL D_TAX;
    neworder_item item[15];
    ACID_STUFF;
} neworder_trans;

```

```

typedef struct {
    int status;
    LOGICAL byname;
    ID W_ID;
    ID D_ID;
    ID C_ID;
    ID C_D_ID;
    ID C_W_ID;
    MONEY H_AMOUNT;
    TEXT H_DATE[20]; /* date as text field */
    TEXT W_STREET_1[20+1];
    TEXT W_STREET_2[20+1];
    TEXT W_CITY[20+1];
    TEXT W_STATE[2+1];
    TEXT W_ZIP[9+1];
    TEXT D_STREET_1[20+1];
    TEXT D_STREET_2[20+1];
    TEXT D_CITY[20+1];
    TEXT D_STATE[2+1];
    TEXT D_ZIP[9+1];
    TEXT C_FIRST[16+1];
    TEXT C_MIDDLE[2+1];
    TEXT C_LAST[16+1];
    TEXT C_STREET_1[20+1];
    TEXT C_STREET_2[20+1];
    TEXT C_CITY[20+1];
    TEXT C_STATE[2+1];
    TEXT C_ZIP[9+1];
    TEXT C_PHONE[16+1];
    TEXT C_SINCE[20]; /* date as text field */
    TEXT C_CREDIT[2+1];
    MONEY C_CREDIT_LIM;
    REAL C_DISCOUNT;
    REAL C_BALANCE;
    TEXT C_DATA[200+1];
    ACID_STUFF;
} payment_trans;

```

```

typedef struct {
    int status;
    LOGICAL byname;
    ID W_ID;
    ID D_ID;
    ID C_ID;
    TEXT C_FIRST[16+1];
    TEXT C_MIDDLE[2+1];
    TEXT C_LAST[16+1];
    MONEY C_BALANCE;
    ID O_ID;
    TEXT O_ENTRY_DATE[20]; /* date as text field */
    ID O_CARRIER_ID;
    COUNT ol_cnt;
    struct {
        ID OL_SUPPLY_W_ID;
        ID OL_I_ID;
        COUNT OL_QUANTITY;
        MONEY OL_AMOUNT;
        TEXT OL_DELIVERY_DATE[20]; /* date as text field */
    } item[15];
    ACID_STUFF;
} ordstat_trans;

```

```

} item[15];
ACID_STUFF;
} ordstat_trans;

```

```

typedef struct {
    int status;
    ID W_ID;
    ID D_ID;
    COUNT threshold;
    COUNT low_stock;
    ACID_STUFF;
} stocklev_trans;

```

```

typedef struct {
    int status;
    ID W_ID;
    ID O_CARRIER_ID;
    struct {
        ID O_ID;
        int status;
    } order[10];
    struct timeval enqueue[1];
    struct timeval deque[1];
    struct timeval complete[1];
    ACID_STUFF;
} delivery_trans;

```

```

typedef union {
    neworder_trans neworder;
    payment_trans payment;
    ordstat_trans ordstat;
    delivery_trans delivery;
    stocklev_trans stocklev;
    int status;
} generic_trans;

```

```

/*****
Record formats for results
*****/

```

```

#ifndef NOTYET
typedef struct
{
    float t1, t2, t3, t4, t5;
    int status :8;
    unsigned int type :3;
    unsigned int ol_cnt :4;
    unsigned int remote_ol_cnt :4;
    unsigned int byname :1;
    unsigned int remote :1;
    unsigned int skipped :4;
} success_t;
#endif

```

```

typedef struct
{
    TIME t1, t2, t3, t4, t5;
    int status;
    unsigned int type :3;
    unsigned int ol_cnt :4;
    unsigned int remote_ol_cnt :4;
    unsigned int byname :1;
    unsigned int remote :1;
    unsigned int skipped :4;
} success_t;

```

```

typedef struct
{
    struct timeval start_time;
} success_header_t;

```

```

/*****
Record formats for loading routines. (DB's have own internal formats)
*****/

```

\*\*\*

```
typedef struct
{
  ID W_ID;
  TEXT W_NAME[10+1];
  TEXT W_STREET_1[20+1];
  TEXT W_STREET_2[20+1];
  TEXT W_CITY[20+1];
  TEXT W_STATE[2+1];
  TEXT W_ZIP[9+1];
  REAL W_TAX;
  MONEY W_YTD;
} warehouse_row;
```

```
typedef struct
{
  ID D_ID;
  ID D_W_ID;
  TEXT D_NAME[10+1];
  TEXT D_STREET_1[20+1];
  TEXT D_STREET_2[20+1];
  TEXT D_CITY[20+1];
  TEXT D_STATE[2+1];
  TEXT D_ZIP[9+1];
  REAL D_TAX;
  MONEY D_YTD;
  ID D_NEXT_O_ID;
} district_row;
```

```
typedef struct
{
  ID C_ID;
  ID C_D_ID;
  ID C_W_ID;
  TEXT C_FIRST[16+1];
  TEXT C_MIDDLE[2+1];
  TEXT C_LAST[16+1];
  TEXT C_STREET_1[20+1];
  TEXT C_STREET_2[20+1];
  TEXT C_CITY[20+1];
  TEXT C_STATE[2+1];
  TEXT C_ZIP[9+1];
  TEXT C_PHONE[16+1];
  DATE C_SINCE;
  TEXT C_CREDIT[2+1];
  MONEY C_CREDIT_LIM;
  REAL C_DISCOUNT;
  MONEY C_BALANCE;
  MONEY C_YTD_PAYMENT;
  COUNT C_PAYMENT_CNT;
  COUNT C_DELIVERY_CNT;
  TEXT C_DATA[500+1];
} customer_row;
```

```
typedef struct
{
  ID H_C_ID;
  ID H_C_D_ID;
  ID H_C_W_ID;
  ID H_D_ID;
  ID H_W_ID;
  DATE H_DATE;
  MONEY H_AMOUNT;
  TEXT H_DATA[24+1];
} history_row;
```

```
typedef struct
{
  ID NO_O_ID;
  ID NO_D_ID;
  ID NO_W_ID;
} neworder_row;
```

```
typedef struct
{
  ID O_ID;
  ID O_D_ID;
  ID O_W_ID;
```

```
  ID O_C_ID;
  DATE O_ENTRY_D;
  ID O_CARRIER_ID;
  COUNT O_OL_CNT;
  LOGICAL O_ALL_LOCAL;
} order_row;
```

```
typedef struct
{
  ID OL_O_ID;
  ID OL_D_ID;
  ID OL_W_ID;
  ID OL_NUMBER;
  ID OL_I_ID;
  ID OL_SUPPLY_W_ID;
  DATE OL_DELIVERY_D;
  COUNT OL_QUANTITY;
  MONEY OL_AMOUNT;
  TEXT OL_DIST_INFO[24+1];
} orderline_row;
```

```
typedef struct
{
  ID I_ID;
  ID I_IM_ID;
  TEXT I_NAME[24+1];
  MONEY I_PRICE;
  TEXT I_DATA[50+1];
} item_row;
```

```
typedef struct
{
  ID S_I_ID;
  ID S_W_ID;
  COUNT S_QUANTITY;
  TEXT S_DIST_01[24+1];
  TEXT S_DIST_02[24+1];
  TEXT S_DIST_03[24+1];
  TEXT S_DIST_04[24+1];
  TEXT S_DIST_05[24+1];
  TEXT S_DIST_06[24+1];
  TEXT S_DIST_07[24+1];
  TEXT S_DIST_08[24+1];
  TEXT S_DIST_09[24+1];
  TEXT S_DIST_10[24+1];
  COUNT S_YTD;
  COUNT S_ORDER_CNT;
  COUNT S_REMOTE_CNT;
  TEXT S_DATA[50+1];
} stock_row;
```

```
/* Empty field values */
#define EMPTY_NUM (MAXINT-1)
#define INVALID_NUM (MAXINT)
#define EMPTY_FLT (MAXDOUBLE)
#define INVALID_FLT (MINDOUBLE)
```

```
/* Status conditions */
#define OK 0
#define E 1
#define E_INVALID_ITEM 2
#define E_NOT_ENOUGH_ORDERS 3
#define E_DB_ERROR 4
#define E_INVALID_INPUT 5
```

```
/* Error message strings */
static char *e_msg[]={"Transaction complete.", "Error", "Invalid item number.",
  "Not enough orders.", "Database ERROR !!!!"};
```

```
#define YES 1
#define NO 0
```

```
double cvt_ftf();
double cvt_money();
```

```

TIME getclock();
TIME getlocalclock();

#define TPC_MSG_QUE 150

/*****
Transaction specific stuff
*****/

/* types of transactions */
#define NEWORDER 1
#define PAYMENT 2
#define ORDSTAT 3
#define DELIVERY 4
#define STOCKLEV 5
#define DEFERRED 6 /* deferred portion of delivery */

/* the name of each transaction */
static char *transaction_name[] =
    { "", "New_Order", "Payment", "Order-Status",
      "Delivery", "Stock-Level", "Deferred-Delivery" };

/* size of each transaction record */
static int transaction_size[] = { 0,
    sizeof(neworder_trans),
    sizeof(payment_trans),
    sizeof(ordstat_trans),
    sizeof(delivery_trans),
    sizeof(stocklev_trans),
    sizeof(delivery_trans),
    0 };

/* valid response time for each transaction */
static TIME valid_response[] = { 0, 5, 5, 5, 5, 20 };

#endif /* TPCC_INCLUDED */

```

---



---

## *lib/date.c*

---



---

```

/*****
*****
@(#) Version: A.10.10 $Date: 97/12/15 10:56:52 $

(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****/
#include "tpcc.h"
#include <time.h>

/* macro to get starting day of a particular year (1901 thru 2100) */
#define YEAR(yr) ((yr-1900)*365 + (yr-1900-1)/4)

CurrentDate(date)
/*****
**
CurrentDate fetches the current date and time
*****/
DATE *date;
{
    struct timeval time;
    struct timezone tz;

    /* get the current time of day */
    if (gettimeofday(&time, &tz) < 0)
        syserror("Can't get time of day\n");

    /* adjust the time of day by the timezone */
    time.tv_sec -= tz.tz_minuteswest * 60;

```

```

/* convert seconds and days since EPOCH (Jan 1, 1970) */
date->day = time.tv_sec / (24*60*60);
date->sec = time.tv_sec - date->day * (24*60*60);

/* convert to days since Jan 1, 1900 */
date->day += YEAR(1970);
}

```

```

EmptyDate(date)
/*****
Get a NULL date and time
*****/
DATE *date;
{
    date->day = 0; /* Use EMPTYNUM instead */
    date->sec = 0;
}

```

```

int IsEmptyDate(date)
DATE *date;
{
    return (date->day == 0 & date->sec == 0);
}

```

```

#define Feb29 (31+29-1)

```

```

fmt_date(str, date)
/*****
**
fmt_date formats the DATE into a string MM-DD-YY HH-MM-SS
*****/
char str[20];
DATE *date;
{
    /* Note: should probably do date and time separately */

```

```

int quad, year, month, day;
int hour, minute, sec;

static int dur[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
static int first = YES;

```

```

day = date->day;
sec = date->sec;

```

```

/* if NULL date, then return empty string */
if (day == EMPTY_NUM || sec == EMPTY_NUM)
    {str[0] = '\0'; return;}

```

```

/* 2100, 1900 are NOT leap years. If we are Feb 29 or later, add a day */
if (day >= Feb29 + YEAR(2100)) day++;
if (day >= Feb29) day++;

```

```

/* figure out which quad and day within quad we are in */
quad = day / (4*365+1);
day = day - quad * (4*365+1);

```

```

/* get our year within quad and day within the year */
if (day < 1*365+1) {year = 0;}
else if (day < 2*365+1) {year = 1; day -= 1*365+1;}
else if (day < 3*365+1) {year = 2; day -= 2*365+1;}
else {year = 3; day -= 3*365+1;}

```

```

/* if this is a leap year, february has 29 days */
if (year == 0) dur[1] = 29;
else dur[1] = 28;

```

```

/* decide which day and month we are */
for (month = 0; day >= dur[month]; month++)
    day -= dur[month];

```

```

/* decide what time of day it is */
minute = sec / 60;
sec = sec - minute * 60;
hour = minute / 60;

```

```

minute = minute - hour * 60;

/* format the date and time */
fmtint(str+0, day+1, 2, ' ');
str[2]='-';
fmtint(str+3, month+1, 2, '0');
str[5]='-';
fmtint(str+6, 1900+quad*4+year, 4, '0');
str[10] = ' ';
fmtint(str+11, hour, 2, ' ');
str[13] = ':';
fmtint(str+14, minute, 2, '0');
str[16] = ':';
fmtint(str+17, sec, 2, '0');
str[19] = '\0';
}

```

---



---

## lib/errlog.c

---



---

```

/*****
*****
@(#) Version: A.10.10 $Date: 97/12/15 10:56:52 $

(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****
*****/
#include<fcntl.h>
#include<stdio.h>
#include<unistd.h>
#include<errno.h>

#include <stdarg.h>
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
extern int errno;

int userid;

error(format, args)
/*****
*****/
error formats a message and outputs it to a standard location (stderr for now)
*****/
char *format;
int args;
{
va_list argptr;

/* point to the list of arguments */
va_start(argptr, args);

/* format and print to stderr */
vmessage(format, argptr);

/* done */
va_end(argptr);

/* take an error exit */
exit(1);
}

syserror( format, args )
/*****
*****/
syserror logs a message with the system error code
*****/
char *format;

```

```

int args;
{
va_list argptr;
int save_errno = errno;

/* point to the list of arguments */
va_start(argptr, args);

/* format and print to stderr */
vmessage(format, argptr);

/* done */
va_end(argptr);

/* display the system error message */
message(" System error message: %s\n", strerror(save_errno));

/* take an error exit */
exit(1);
}

```

```

message(format, args)
/*****
*****/

```

```

message formats a message and outputs it to a standard location (stderr for now)
*****/

```

```

char *format;
int args;
{
va_list argptr;

/* point to the list of arguments */
va_start(argptr, args);

/* format and print to stderr */
vmessage(format, argptr);

/* done */
va_end(argptr);
}

```

```

vmessage(format, argptr)
/*****
*****/

```

```

*****/
char *format;
va_list argptr;
{
char buf[3*1024];

/* format a message id */
sprintf(buf, "User %-6d Pid %-6d ", userid, getpid());

/* format the string and print it */
vsprintf(buf+strlen(buf), format, argptr);
if (getenv("NO_ERROR_LOG") == NULL)
msg_buf(buf, strlen(buf));
if (getenv("NO_STDERR") == NULL)
write(2, buf, strlen(buf));
}

```

```

static msg_buf(buf, size)
char *buf;
int size;
{
int fd;
char *fname;

```

```

/* get the file name to use */
fname = getenv("ERROR_LOG");
if (fname == NULL)
    fname = "/tmp/ERROR_LOG";

/* get exclusive access to the error log file */
fd= open(fname, O_WRONLY | O_CREAT, 0666);
if (fd < 0)
    console_error("Can't open tpc error log file 'ERROR_LOG'\n");
lockf(fd, F_LOCK, 0);

/* write the new text at the end of the file */
lseek(fd, 0, SEEK_END);
write(fd, buf, size);

/* release the file */
/* fsync(fd); */
lockf(fd, F_ULOCK, 0);
close(fd);
}

```

```

console_error(str)
char *str;
{
    int fd = open("/dev/tty", O_WRONLY);
    write(fd, str, strlen(str));
    close(fd);
    exit(1);
}

```

---



---

## lib/fmt.c

---



---

```

*****
*****

```

```
@(#) Version: A.10.10 $Date: 97/12/15 10:56:52 $
```

```
(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
```

```
*****
*****/
```

```

#include "tpcc.h"
#include "iobuf.h"
#include <math.h> /* needed for ceil (VM) */
#include <strings.h>

```

```
/* formatting routines. */
```

```
/* Note: Currently use integer routines to format and convert. Need to
modify the code for cases when integers don't work. */
```

```

fmt_money(str, m, width)
char *str;
MONEY m;
int width;
{
    if (m == EMPTY_FLT)
    {
        memset(str, '_', width);
        str[width] = '\0';
        return;
    }

    /* format it as a number with a leading blank */
    *str = ' ';
    fmt_flt(str+1, m/100, width-1, 2);

```

```

/* fill in a leading dollar */
while (*(str+1) == ' ')
    str++;
*str = '$';

```

```

}

double cvt_money(str)
char *str;
{
    char temp[81], *t, *s;
    double cvt_flt(), f;

    /* skip leading and trailing blanks */
    cvt_text(str, temp);

    /* remove leading $ */
    if (*temp == '$') t = temp + 1;
    else t = temp;

    /* start scan at current character */
    s = t;

    /* allow leading minus sign */
    if (*s == '-')
        s++;

    /* allow leading digits */
    while (isdigit(*s))
        s++;

    /* allow decimal pt and two decimal digits */
    if (*s == '.') s++;
    if (isdigit(*s)) s++;
    if (isdigit(*s)) s++;

    /* There should be no more characters */
    if (*s != '\0') return INVALID_FLT;

    /* convert the floating pt number */
    f = cvt_flt(t);
    if (f == EMPTY_FLT) return EMPTY_FLT;
    else if (f == INVALID_FLT) return INVALID_FLT;
    else return rint(f*100);
}

```

```

}

```

```

fmt_num(str, n, width)
char str[];
int n;
int width;
{
    /* mark the end of the string */
    str[width] = '\0';

    /* if empty number, return the empty field */
    if (n == EMPTY_NUM)
        memset(str, '_', width);

    /* otherwise, convert the integer */
    else
        fmtint(str, n, width, ' ');

    debug("fmt_num: n=%d str=%s\n", n, str);
}

```

```

cvt_num(str)
char str[];
{
    char text[81];
    cvt_text(str, text);
    if (*text == '\0')
        return EMPTY_NUM;
    else
        return cvtint(text);
}

```

```

fmt_flt(str, x, width, dec)
*****

```



```

****
fmt_ft converts a floating pt number to a string "999999.9999"
*****
****/
char *str;
double x;
int width;
int dec;
{
int negative;
int integer, fract;
double absolute;

static double pow10[] =
{1., 10., 100., 1000., 10000., 100000., 1000000., 10000000., 100000000.};

/* mark the end of string */
str[width] = '\0';

/* if empty value, make it be an empty field */
if (x == EMPTY_FLT)
{
memset(str, '_', width);
return;
}

absolute = (x < 0)? -x: x;

/* separate into integer and fractional parts */
integer = (int) absolute;
fract = (absolute - integer) * pow10[dec] + .5;

/* let the integer portion contain the sign */
if (x < 0) integer = -integer;

/* Format integer and fraction separately */
fmtint(str, integer, width-dec-1, '.');
str[width-dec-1] = '.';
fmtint(str+width-dec, fract, dec, '0');
}

double cvt_ft(str)
char str[];
{
char text[81];
char *t;
double value;
int div;
int fract;
int negative;
int i;

/* normalize the text */
cvt_text(str, text);
if (*text == '\0')
return EMPTY_FLT;

negative = NO;
fract = NO;
value = 0;
div = 1.0;

negative = (text[0] == '-');
if (negative) t = text+1;
else t = text;

for (; *t != '\0' ; t++)
{
if (*t == '.')
if (fract) return INVALID_FLT;
else fract = YES;

else if (isdigit(*t))
{
value = value*10 + (int)*t - (int)'0';
}
}
}

```

```

if (fract) div *= 10;
}

else
return INVALID_FLT;
}

if (fract)
value /= div;

if (negative)
value = -value;

return value;
}

fmt_text(s, text, width)
char *s, *text;
int width;
{

/* if an empty string, then all underscores */
if (*text == '\0')
for (; width > 0; width--)
*s++ = '_';

/* otherwise, blank fill it */
else
{

/* copy the text into the new buffer */
for (; *text != '\0'; width--)
*s++ = *text++;

/* fill in the rest with blanks */
for (; width > 0; width--)
*s++ = ' ';

}

/* and finally, terminate the string */
*s = '\0';
}

cvt_text(s, text)
char *s;
char *text;
{
char *lastnb;

/* skip leading blanks and underscores */
for (; *s == ' ' || *s == '_'; s++)
;

/* copy the characters, keeping track of last blank or underscore */
lastnb = text-1;
for (; *s != '\0'; *text++ = *s++)
if (*s != ' ' && *s != '_')
lastnb = text;

/* truncate the text string to last nonblank character */
*(lastnb+1) = '\0';
}

fmtint(field, value, size, fill)
/*****
fmtint formats an integer value into a character field to make the integer
right-justified within the character field, padded with leading fill
characters (e.g. leading blanks if a blank is passed in for the fill argument
*****/
int value;
char *field;

```

```

int size;
char fill;
{
int negative;
int dividend;
int remainder;
char *p;

/* create characters from right to left */
p = field + size - 1;

/* make note if this is a negative number */
negative = value < 0;
if (negative)
    value = -value;

/* Case: Null field. Can't do anything */
if (p < field)
    ;

/* Case: value is zero. Print a leading '0' */
else if (value == 0)
    *p-- = '0';

/* Otherwise, convert each digit in turn */
else do
    {

        dividend = value / 10;
        remainder = value - dividend * 10;
        value = dividend;

        *p-- = (char) ( (int)'0' + remainder );

    } while (p >= field && value > 0);

/* insert a minus sign if appropriate */
if (negative && p >= field)
    *p-- = '-';

/* fill in leading characters */
while (p >= field)
    *p-- = fill;
}

```

```

int cvtint(str)
/******
getint extracts an integer value from the given character field
(ex: turns the string "123" into the integer 123)
******/
char *str;
{
int value;
char c;
int negative;
debug("cvtint: str=%s\n", str);

negative = (*str == '-');
if (negative) str++;

/* convert the integer */
for (value = 0; isdigit(*str); str++)
    value = value*10 + (int)(*str) - (int)'0';

/* if any non-digit characters, error */
if (*str != '\0')
    return INVALID_NUM;

/* make negative if there was a minus sign */
if (negative)
    value = -value;

debug("cvtint: value=%d\n", value);
return value;
}

```

```

fmt_phone(str, phone)
char str[20];
char *phone;

{
/* copy phone number and insert dashes 999999-999-999-9999 */
str[0] = phone[0]; str[1] = phone[1]; str[2] = phone[2];
str[3] = phone[3]; str[4] = phone[4]; str[5] = phone[5];
str[6] = '-';
str[7] = phone[6]; str[8] = phone[7]; str[9] = phone[8];
str[10] = '-';
str[11] = phone[9]; str[12] = phone[10]; str[13] = phone[11];
str[14] = '-';
str[15] = phone[12]; str[16] = phone[13]; str[17] = phone[14];
str[18] = phone[15];
str[19] = '\0';
}

```

```

fmt_zip(str,zip)
char str[20];
char *zip;
{
/* copy zip code and insert dashes 99999-9999 */
str[0] = zip[0]; str[1] = zip[1]; str[2] = zip[2];
str[3] = zip[3]; str[4] = zip[4];
str[5] = '-';
str[6] = zip[5]; str[7] = zip[6]; str[8] = zip[7]; str[9] = zip[8];
str[10] = '\0';
}

```

---



---

## lib/iobuf.h

---



---

```

/******
*****
@(#) Version: A.10.10 $Date: 97/12/15 10:56:52 $

```

```

(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****
*****

```

```

/******
*
History
941220 LAN Added definition and initialization of the line_col[] array.
This was needed for modifications made of client program to do
block I/O using a WYSE terminal.
*****
/

```

```

/* structure for screen emulation */
typedef struct
{
int row;
int col;
char buf[25][81];
} screen_t;

```

```

typedef struct {
char *beg;
char *end; /* for output buffers */
char *max;
char *cur; /* for input buffers */
} iobuf;

```

```

/* Macro do define an I/O buffer of x characters, initialized to empty */
#define define_iobuf(name, size) \
char name/**/_data[size]; \
iobuf name[1] = { {name/**/_data, name/**/_data, \
name/**/_data+size, name/**/_data} }

```

```

#define reset(buf) if (1) { \
    (buf)->cur = (buf)->end = (buf)->beg; \
    *(buf)->beg = '\0'; \
} else (void)0

#define flush() if(1) { \
    display(out_buf); \
    reset(out_buf); \
} else (void)0

/* Standard I/O to and from in_buf and out_buf */
#ifndef DECLARE_IO_BUFFERS
#define _iobuf(output_stuff, 4*1024);
#define _iobuf(input_stuff, 1024);
_iobuf *in_buf = input_stuff;
_iobuf *out_buf = output_stuff;
#else
_iobuf *in_buf;
_iobuf *out_buf;
#endif

#define pushc(c) if (1) { \
    if (out_buf->end >= out_buf->max) \
        error("out_buf overflow: beg=0x%x end=%d max=%d\n", \
            out_buf->beg, out_buf->end-out_buf->beg, out_buf->max-out_buf->beg); \
    *(out_buf->end++) = (c); \
    *(out_buf->end) = '\0'; /* debug */ \
} else (void)0

#define popc() \
    (*(in_buf->cur++)

/* Standard characters used for screen control */
#define ENTER '\015'
#define TAB '\t'
#define BACKTAB '\02' /* ^B */
#define CNTRL '\03'
#define BACKSPACE '\010'
#define BELL '\007'
#define BLANK ' '
#define UNDERLINE '_'
#define ESCAPE '\033'
/*#define EOF ((char)-1) */
#define TRIGGER '\021' /* dc1 */

```

---



---

## *lib/iobuf.c*

---



---

```

/*****
*****
@(#) Version: A.10.10 $Date: 97/12/15 10:56:52 $

(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****/
#define DECLARE_IO_BUFFERS
#include "iobuf.h"
#undef DECLARE_IO_BUFFERS
#include "tpcc.h"
extern int errno;

```

```

string(str)
char str[];
{
    for (; *str != '\0'; str++)
        pushc(*str);
}

```

```

}

push(str, len)
char *str;
int len;
{
    for (; len > 0; len --)
        pushc(*str++);
}

display(scr)
_iobuf *scr;
{
    /* Note: if problems doing output, let the input routine detect it */
    char *p;
    int len;
    for (p = scr->beg; p < scr->end; p+=len)
    {
        len = write(1, p, scr->end - p);
        if (len <= 0) break;
    }
}

input(scr)
_iobuf *scr;
{
    int len;

    /* read in as many characters as are available */
    len = read(0, scr->end, scr->max - scr->end);

    /* if end of input, then pretend we read an END character */
    if (len == 0 || (len == -1 && errno == ECONNRESET))
    {
        *scr->end = EOF;
        len = 1;
    }

    /* Check for errors */
    else if (len == -1)
        syserror("input(scr): unable to read stdin\n");

    /* update the pointers to reflect the new data */
    scr->end += len;
    *scr->end = '\0'; /* for debugging */
}

```

```

getkey()
{
    if (in_buf->cur == in_buf->end)
    {
        flush();
        reset(in_buf);
        input(in_buf);
    }

    return popc();
}

```

---



---

## *lib/random.c*

---



---

```

/*****
*****
@(#) Version: A.10.10 $Date: 97/12/15 14:01:59 $

```

(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.

```
*****
*****/
#include "tpcc.h"
#include "string.h"
#include "random.h"

double drand48();

char lastNames[1000][16];
char customerData1[10][301];
char customerData2[10][201];
char stockData1[10][27];
char stockData2[10][25];
char historyData1[10][13];
char historyData2[10][13];
char citystreetData1[10][11];
char citystreetData2[10][11];
char firstNameData1[10][9];
char firstNameData2[10][9];
char StockDistrict[10][25];
char phoneData[10][17];

static long RandySeedIter = 7;

void GenerateLastNames()
{
    int i;
    char *name;
    static char *n[] = {"BAR", "OUGHT", "ABLE", "PRI", "PRES",
                       "ESE", "ANTI", "CALLY", "ATION", "EING"};

    for(i = 0; i < 1000; i++) {
        name = lastNames[i];
        strcpy(name, n[(i/100)%10]);
        strcat(name, n[(i/10) %10]);
        strcat(name, n[(i/1) %10]);
    }
}

int MakeNumberString(min, max, num)
int min;
int max;
TEXT num[];
{
    static char digit[]="0123456789";
    int length;
    int i;

    length = RandomNumber(min, max);

    for (i=0; i<length; i++)
        num[i] = digit[RandomNumber(0,9)];
    num[length] = '\0';

    return length;
}

ID RandomWarehouse(local, scale, percent)
ID local;
ID scale;
int percent; /* percent of remote transactions */
{
    ID w_id;

    /* For the given percent of the time, pick the local warehouse */
    if (RandomNumber(1, 100) > percent || scale == 1)
        w_id = local;

    /* Otherwise, pick a non-local warehouse */
    else
    {
        w_id = RandomNumber(2, scale);
        if (w_id == local)
            w_id = 1;
    }
    return w_id;
}
```

```
/* Initialize a table of Random strings for the stock-district
field in the stock table. We can use a table of 10 elements
and select randomly from this table via rule 4.3.2.2 in
the TPC-C spec */
```

```
void InitRandomStrings()
```

```
{
    int i;

    for (i=0; i < 10; i++) {
        MakeAlphaString(24,24,&StockDistrict[i]);

        MakeAlphaString(300,300,&customerData1[i]);
        MakeAlphaString(0,200,&customerData2[i]);

        MakeAlphaString(26,26,&stockData1[i]);
        MakeAlphaString(0,24,&stockData2[i]);

        MakeAlphaString(12,12,&historyData1[i]);
        MakeAlphaString(0,12, &historyData2[i]);

        MakeAlphaString(10,10,&citystreetData1[i]);
        MakeAlphaString(0,10,&citystreetData2[i]);

        MakeAlphaString(8,8,&firstNameData1[i]);
        MakeAlphaString(0,8,&firstNameData2[i]);

        MakeNumberString(16,16,&phoneData[i]);
    }
    GenerateLastNames();
}
```

```
int MakeAlphaString(min, max, str)
```

```
int min;
int max;
TEXT str[];
{
    static char character[] = "abcdefghijklmnopqrstuvwxy";
    int length;
    int i;
```

```
length = RandomNumber(min, max);
```

```
for (i=0; i<length; i++) {
    /* NOTE: we use sizeof(character)-2 because of the following:
    subtract 1 because we are numbering from 0 instead of 1 and
    subtract 1 because the sizeof(character) is 1 greater than
    the data in character because of the invisible C string
    terminator at the end. */
    str[i] = character[RandomNumber(0, sizeof(character)-2)];
}
str[length] = '\0';

return length;
}
```

```
void RandomPermutation(perm, n)
```

```
int perm[];
int n;
{
    int i, r, t;

    /* generate the identity permutation to start with */
    for (i=1; i<=n; i++)
        perm[i] = i;

    /* randomly shuffle the permutation */
    for (i=1; i<=n; i++)
    {
        r = RandomNumber(i, n);
        t = perm[i]; perm[i] = perm[r]; perm[r] = t;
    }
}
```

```
void RandomDelay(mean, adjust)
```

```
/******
random_sleep sleeps according to the TPC specification
```

```

*****/
double mean;
double adjust;
{
double secs;
double exponential();

secs = exponential(mean);

delay(secs+adjust);
}

double exponential(mean)
/*****
exponential generates a reverse exponential distribution
*****/
double mean;
{
double x;
double log();

#ifdef USE_DRAND48
x = -log(1.0-drand48()) * mean;
#else
x = -log(1.0-randy()) * mean;
#endif

return x;
}

void SetRandomSeed(val)
long val;
{
#ifdef USE_DRAND48
srand48(val);
#else
RandySeedIter = val;
#endif
}

void Randomize()
{
SetRandomSeed(time(0)+getpid());
}

/* Random number generator from Proceeding of the ACM */
#define RANDY_A_VAL 16807
/* 2^31 - 1 */
#define RANDY_M_VAL 2147483647
/* m / a */
#define RANDY_Q_VAL 127773
/* m % a */
#define RANDY_R_VAL 2836

double randy()
{
long hi, lo, test;

hi = RandySeedIter / RANDY_Q_VAL;
lo = RandySeedIter % RANDY_Q_VAL;

test = (RANDY_A_VAL * lo) - (RANDY_R_VAL * hi);
RandySeedIter = (test > 0) ? test : test + RANDY_M_VAL;

return( (double)RandySeedIter / (double)RANDY_M_VAL );
} /* end of fn randy */

```

---

## *lib/random.h*

---

```

*****
*****

```

@(#) Version: A.10.10 \$Date: 97/12/15 14:02:00 \$

```

(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****/
#ifndef TPCC_RANDOM
#define TPCC_RANDOM

#ifdef USE_DRAND48
double drand48();
#else
double randy();
#endif

extern int  MakeNumberString();
extern ID   RandomWarehouse();
extern int  MakeAlphaString();
extern void RandomPermutation();
extern void RandomDelay();
extern double exponential();
extern void Randomize();
extern void SetRandomSeed();

extern char lastNames[1000][16];
extern char customerData1[10][301];
extern char customerData2[10][201];
extern char stockData1[10][27];
extern char stockData2[10][25];
extern char historyData1[10][13];
extern char historyData2[10][13];
extern char citystreetData1[10][11];
extern char citystreetData2[10][11];
extern char firstNameData1[10][9];
extern char firstNameData2[10][9];
extern char StockDistrict[10][25];
extern char phoneData[10][17];

/*****
****
RandomNumber selects a uniform random number from min to max inclusive
*****/
#ifdef USE_DRAND48
#define RandomNumber(min,max) \
((int)(drand48() * ((int)(max) - (int)(min) + 1)) + (int)(min))
#else
#define RandomNumber(min,max) \
((int)(randy() * ((int)(max) - (int)(min) + 1)) + (int)(min))
#endif

/*****
****
NURandomNumber selects a non-uniform random number
*****/
#define NURandomNumber(a, min, max, c) \
((RandomNumber(0, a) | RandomNumber(min, max)) + (c)) % \
((max) - (min) + 1) + (min)

#define SelectCityStreetData(data) \
{ \
strcpy(data,citystreetData1[RandomNumber(0,9)]); \
strcat(data,citystreetData2[RandomNumber(0,9)]); \
}

#define SelectFirstName(data) \
{ \
strcpy(data,firstNameData1[RandomNumber(0,9)]); \
strcat(data,firstNameData2[RandomNumber(0,9)]); \
}

#define SelectHistoryData(data) \
{ \
strcpy(data,historyData1[RandomNumber(0,9)]); \
strcat(data,historyData2[RandomNumber(0,9)]); \
}

#define SelectStockData(data) \
{ \

```

```

    strcpy(data,stockData1[RandomNumber(0,9)]); \
    strcat(data,stockData2[RandomNumber(0,9)]); \
}

#define SelectClientData(data) \
{ \
    strcpy(data,customerData1[RandomNumber(0,9)]); \
    strcat(data,customerData2[RandomNumber(0,9)]); \
}

#define SelectPhoneData(data) strcpy(data,phoneData[RandomNumber(0,9)])
#define SelectStockDistrict(data) strcpy(data,StockDistrict[RandomNumber(0,9)])

#define MakeZip(zip) \
{ \
    MakeNumberString(4, 4, zip); \
    zip[4] = '1'; \
    zip[5] = '1'; \
    zip[6] = '1'; \
    zip[7] = '1'; \
    zip[8] = '1'; \
    zip[9] = '\0'; \
}

#define MakeAddress(str1, str2, city, state, zip) \
{ \
    SelectCityStreetData(str1); \
    SelectCityStreetData(str2); \
    SelectCityStreetData(city); \
    MakeAlphaString(2,2,state); \
    MakeZip(zip); \
}

#define LastName(num, name) strcpy(name, lastNames[(num)])

#define Original(str) \
{ \
    int len = strlen(str); \
    if (len >= 8) { \
        int pos = RandomNumber(0,(len-8)); \
        str[pos+0] = 'O'; \
        str[pos+1] = 'R'; \
        str[pos+2] = 'I'; \
        str[pos+3] = 'G'; \
        str[pos+4] = 'T'; \
        str[pos+5] = 'N'; \
        str[pos+6] = 'A'; \
        str[pos+7] = 'L'; \
    } \
}

#endif

```

---



---

## *lib/results\_file.c*

---



---

```

/*****
*****
@(#) Version: A.10.10 $Date: 97/12/15 14:02:01 $

(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****/
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include "tpcc.h"

```

```

static FILE *rfile;

results_open(id)
int id;
{
    char fullname[128];
    char *basename;

    /* get the base file name for the deferred results */
    /*
    * Make it a directory under /tmp so at least we can set it to a
    * symbolic link in case /tmp doesn't have enough room.
    */
    basename = getenv("TPCC_RESULTS_FILE");
    if (basename == NULL)
        basename = "/tmp/TPCC_RESULTS_FILE";

    /* create the full file name */
    sprintf(fullname, "%s.%d", basename, id);

    /* open the file */
    unlink(fullname);
    rfile = fopen(fullname, "wb");
    if (rfile == NULL)
        syserror("Delivery server %d can't open file %s\n", id, fullname);

    /* allocate a larger buffer */
}

results(t)
delivery_trans *t;
{
    if (fwrite(t, sizeof(*t), 1, rfile) != 1)
        syserror("Delivery server: Can't post results\n");
}

results_close()
{
    if (fclose(rfile) < 0)
        syserror("Delivery server can't close file\n");
}

```

---



---

## *lib/Makefile*

---



---

```

*****
*****
@(#) Version: A.10.10 $Date: 97/12/15 14:02:02 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****
*****

debug = +O4 +Ofastaccess +Oentrysched
#debug = -g

CFLAGS= ${debug} -Wl,-a,archive -I. +ESlit

utils=iobuf.o delay.o errlog.o fmt.o random.o tas.o null_key.o null_select.o
results_file.o date.o prepare_socket.o shm.o spinlock.o

all: tpc_lib.a server_default.o

tpc_lib.a: ${utils}
    rm -f tpc_lib.a
    ar -r tpc_lib.a ${utils}

clean:
    rm -f *.o

```

```

rm -f *.a

clobber: clean

.s.o:
cc -c $*.s

```

```

dbexit();
}

#define INT2(p) ((short *(p)+1)
#define INT1(p) ((char *(p)+3)

void neworder_transaction(t)
neworder_trans *t;
{
int try;

/* repeat until we give up trying */
for (try=0; try<MaxTries; try++)
{
/* if the transaction succeeds, then done */
if (neworder_body(t))
break;

/* clean up and try again */
dbcancel(dbproc);
sleep_before_retry();

/* don't retry if caused by operator error */
if (t->status == E_INVALID_INPUT) break;
}

/* if we finally gave up, then display a message */
if (try >= MaxTries)
t->status = E_DB_ERROR;
}

int neworder_body (t)
neworder_trans *t;
{
int i;
DBDATETIME o_entry_d;
DATE o_entry_d_DATE;
REAL tax_n_discount;

debug("Neworder: w_id=%d d_id=%d c_id=%d\n", t->W_ID, t->D_ID, t->C_ID);

/* assume everthing fine unless otherwise */
t->status = OK;

/* see if our items are all local */
for (i=0; i<t->O_OL_CNT; i++)
if (t->item[i].OL_SUPPLY_W_ID != t->W_ID) break;
t->all_local = (i >= t->O_OL_CNT);

/* prepare the parameters for the "neworder" transaction. */
if (t->all_local) dbrpcinit(dbproc, "neworder_local", 0);
else dbrpcinit(dbproc, "neworder_remote", 0);
dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->W_ID));
dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&t->D_ID));
dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, &t->C_ID);
dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, &t->O_OL_CNT);

/* Send the orderlines (up to 15) */
for (i = 0; i < t->O_OL_CNT; i++)
{
debug(" i=%d i_id=%d w_id=%d qty=%d\n", i, t->item[i].OL_I_ID,
t->item[i].OL_SUPPLY_W_ID, t->item[i].OL_QUANTITY);
dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, &t->item[i].OL_I_ID);
if (!t->all_local)
dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1,
INT2(&t->item[i].OL_SUPPLY_W_ID));
dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&t-
>item[i].OL_QUANTITY));
}

/* execute the neworder transaction */
if (dbrpcsend(dbproc) != SUCCEED) return NO;
}

```

## A.3 Transaction Source

### *client/sybase/transaction.c*

```

/*****
*****
@(#) Version: A.10.10 $Date: 97/09/11 11:26:36 $

(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****/

#include <sybfront.h>
#include <sybdb.h>
#include "tpcc.h"

#define MaxTries 10

int user;
LOGINREC *login;
DBPROCESS *dbproc;

#define from_sybase_date(sybddate, date) { \
date.day = sybddate.dtdays; \
date.sec = sybddate.dtime / 300; \
}

transaction_begin(u)
int u;
{
char *packet;
int message_handler(), error_handler();

user = u;

/* initialize dblib */
if (dbinit() != SUCCEED)
error("Can't initialize the DB library\n");

/* install a message handler */
(void)dbmsghandle(message_handler);
(void)dberrhandle(error_handler);

/* set up Sybase structures */
login = dblogin();
DBSETLUSER(login, "sa");
DBSETLPACKET(login, 4096);

/* Open the connection to the server. */
if ((dbproc = dbopen(login, (char *)NULL)) == NULL)
error("Could not open connection\n");

/* Use the TPCC database */
dbuse(dbproc, "tpcc");
}

transaction_done()
{
/* put detach from database here */
}

```

```

if (dbsqlok(dbproc) != SUCCEED) return NO;

/* get results from order lines */
for (i = 0; i < t->O_OL_CNT; i++)
    if (!order_line_result(dbproc, &t->item[i], &t->status))
        break;

/* get the results of the overall neworder transaction */
if (dbresults(dbproc) != SUCCEED) return NO;
dbbind(dbproc, 1, FLT8BIND, 0, &t->W_TAX);
dbbind(dbproc, 2, FLT8BIND, 0, &t->D_TAX);
dbbind(dbproc, 3, INTBIND, 0, &t->O_ID);
dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(t->C_LAST), t->C_LAST);
dbbind(dbproc, 5, FLT8BIND, 0, &t->C_DISCOUNT);
dbbind(dbproc, 6, NTBSTRINGBIND, sizeof(t->C_CREDIT), t->C_CREDIT);
dbbind(dbproc, 7, DATETIMEBIND, 0, &o_entry_d);
if (dbnextrow(dbproc) != REG_ROW) return NO;
if (dbcquery(dbproc) != SUCCEED) return NO;

/* convert the date */
from_sybase_date(o_entry_d, o_entry_d_DATE);
fmt_date(&t->O_ENTRY_D, &o_entry_d_DATE);

/* Check for invalid input (what is -6 anyway?) */
if (dbretstatus(dbproc) == -6) {
    t->status = E_INVALID_INPUT;
    return NO;
}
/* done */
return YES;
}

int order_line_result(dbproc, item, status)
DBPROCESS *dbproc;
neworder_item *item;
int *status;
{
    /* Each order line is a separate query. Fetch the data */
    if (dbresults(dbproc) != SUCCEED) return NO;
    dbbind(dbproc, 1, NTBSTRINGBIND, sizeof(item->I_NAME), item->I_NAME);
    dbbind(dbproc, 2, FLT8BIND, 0, &item->I_PRICE);
    dbbind(dbproc, 3, INTBIND, 0, &item->S_QUANTITY);
    dbbind(dbproc, 4, CHARBIND, 1, &item->brand_generic);
    if (dbnextrow(dbproc) != REG_ROW) return NO;
    if (dbcquery(dbproc) != SUCCEED) return NO;
    if (dbhasretstat(dbproc) && dbretstatus(dbproc) != 0) return NO;

    /* Note: items that weren't found will have empty I_NAME */
    if (item->I_NAME[0] == '^0')
        *status = E_INVALID_ITEM;
    return YES;
}

void payment_transaction(t)
payment_trans *t;
{
    int try;

    /* repeat until we give up trying */
    for (try=0; try<MaxTries; try++)
    {
        /* if the transaction succeeds, then done */
        if (payment_body(t))
            break;

        /* clean up and try again */
        dbcancel(dbproc);
        sleep_before_retry();

        /* don't retry if caused by operator error */
        if (t->status == E_INVALID_INPUT) break;
    }

    /* if we finally gave up, then display a message */
    if (try >= MaxTries)
        t->status = E_DB_ERROR;
}

int payment_body(t)
payment_trans *t;
{
    DBDATETIME H_DATE;
    DATE H_DATE_DATE;
    DBDATETIME C_SINCE;
    DATE C_SINCE_DATE;

    if (t->byname)
    {
        dbrpcinit(dbproc, "payment_byname", 0);
        dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->W_ID));
        dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->C_W_ID));
        dbrpcparam(dbproc, NULL, 0, SYBFLT8, -1, -1, &t->H_AMOUNT);
        dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&t->D_ID));
        dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&t->C_D_ID));
        dbrpcparam(dbproc, NULL, 0, SYBCHAR, -1, strlen(t->C_LAST), t->C_LAST);
    }
    else
    {
        dbrpcinit(dbproc, "payment_byid", 0);
        dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->W_ID));
        dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->C_W_ID));
        dbrpcparam(dbproc, NULL, 0, SYBFLT8, -1, -1, &t->H_AMOUNT);
        dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&t->D_ID));
        dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&t->C_D_ID));
        dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, &t->C_ID);
    }

    if (dbrpcsend(dbproc) != SUCCEED) return NO;
    if (dbsqlok(dbproc) != SUCCEED) return NO;
    if (dbresults(dbproc) != SUCCEED) return NO;

    dbbind(dbproc, 1, INTBIND, 0, &t->C_ID);
    dbbind(dbproc, 2, NTBSTRINGBIND, sizeof(t->C_LAST), t->C_LAST);
    dbbind(dbproc, 3, DATETIMEBIND, 0, &H_DATE);
    dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(t->W_STREET_1), t->W_STREET_1);
    dbbind(dbproc, 5, NTBSTRINGBIND, sizeof(t->W_STREET_2), t->W_STREET_2);
    dbbind(dbproc, 6, NTBSTRINGBIND, sizeof(t->W_CITY), t->W_CITY);
    dbbind(dbproc, 7, NTBSTRINGBIND, sizeof(t->W_STATE), t->W_STATE);
    dbbind(dbproc, 8, NTBSTRINGBIND, sizeof(t->W_ZIP), t->W_ZIP);

    dbbind(dbproc, 9, NTBSTRINGBIND, sizeof(t->D_STREET_1), t->D_STREET_1);
    dbbind(dbproc, 10, NTBSTRINGBIND, sizeof(t->D_STREET_2), t->D_STREET_2);
    dbbind(dbproc, 11, NTBSTRINGBIND, sizeof(t->D_CITY), t->D_CITY);
    dbbind(dbproc, 12, NTBSTRINGBIND, sizeof(t->D_STATE), t->D_STATE);
    dbbind(dbproc, 13, NTBSTRINGBIND, sizeof(t->D_ZIP), t->D_ZIP);

    dbbind(dbproc, 14, NTBSTRINGBIND, sizeof(t->C_FIRST), t->C_FIRST);
    dbbind(dbproc, 15, NTBSTRINGBIND, sizeof(t->C_MIDDLE), t->C_MIDDLE);
    dbbind(dbproc, 16, NTBSTRINGBIND, sizeof(t->C_STREET_1), t->C_STREET_1);
    dbbind(dbproc, 17, NTBSTRINGBIND, sizeof(t->C_STREET_2), t->C_STREET_2);
    dbbind(dbproc, 18, NTBSTRINGBIND, sizeof(t->C_CITY), t->C_CITY);
    dbbind(dbproc, 19, NTBSTRINGBIND, sizeof(t->C_STATE), t->C_STATE);
    dbbind(dbproc, 20, NTBSTRINGBIND, sizeof(t->C_ZIP), t->C_ZIP);
    dbbind(dbproc, 21, NTBSTRINGBIND, sizeof(t->C_PHONE), t->C_PHONE);
    dbbind(dbproc, 22, DATETIMEBIND, 0, &C_SINCE);
    dbbind(dbproc, 23, NTBSTRINGBIND, sizeof(t->C_CREDIT), t->C_CREDIT);
    dbbind(dbproc, 24, FLT8BIND, 0, &t->C_CREDIT_LIM);
    dbbind(dbproc, 25, FLT8BIND, 0, &t->C_DISCOUNT);
    dbbind(dbproc, 26, FLT8BIND, 0, &t->C_BALANCE);
    dbbind(dbproc, 27, NTBSTRINGBIND, sizeof(t->C_DATA), t->C_DATA);

    if (dbnextrow(dbproc) != REG_ROW) return NO;
    if (dbcquery(dbproc) != SUCCEED) return NO;

    t->status = OK;
}

```



```

/* need to be in string format */

from_sybase_date(C_SINCE, C_SINCE_DATE);
from_sybase_date(H_DATE, H_DATE_DATE);
fmt_date(t->H_DATE, &H_DATE_DATE);
fmt_date(t->C_SINCE, &C_SINCE_DATE);

/* Check for invalid input */
if (dbretstatus(dbproc) == -6) {
    t->status = E_INVALID_INPUT;
    return NO;
}

return YES;
}

void ordstat_transaction(t)
ordstat_trans *t;
{
    int try;

/* repeat until we give up trying */
for (try=0; try<MaxTries; try++)
    {

/* if the transaction succeeds, then done */
if (ordstat_body(t))
    break;

/* clean up and try again */
dbcancel(dbproc);
sleep_before_retry();

/* don't retry if caused by operator error */
if (t->status == E_INVALID_INPUT) break;
}

/* if we finally gave up, then display a message */
if (try >= MaxTries)
    t->status = E_DB_ERROR;
}

int ordstat_body(t)
ordstat_trans *t;
{
    ID ol_supply_w_id;
    ID ol_i_id;
    COUNT ol_quantity;
    MONEY ol_amount;
    DBDATETIME ol_delivery_d;
    DBDATETIME o_entry_d;
    DATE ol_delivery_d_DATE;
    DATE o_entry_d_DATE;

    int i, code;

/* if this is by name, then invoke the byname procedure */
if (t->byname)
    {
        dbrpcinit(dbproc, "order_status_byname", 0);
        dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->W_ID));
        dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&t->D_ID));
        dbrpcparam(dbproc, NULL, 0, SYBCHAR, -1, strlen(t->C_LAST), t->C_LAST);
    }

/* otherwise, invoke the by id procedure */
else
    {
        dbrpcinit(dbproc, "order_status_byid", 0);
        dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->W_ID));
        dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&t->D_ID));
        dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, &t->C_ID);
    }

/* make the rpc call and check for errors */
if (dbrpcsend(dbproc) != SUCCEED) return NO;
if (dbsqlok(dbproc) != SUCCEED) return NO;
if (dbresults(dbproc) != SUCCEED) return NO;

/* Code for TPC-C rev. 3.3 error checking. */
if (dbrows(dbproc) != SUCCEED) {
    t->status = E_INVALID_INPUT;
}

/* prepare to fetch the results */
dbbind(dbproc, 1, INTBIND, 0, &ol_supply_w_id);
dbbind(dbproc, 2, INTBIND, 0, &ol_i_id);
dbbind(dbproc, 3, INTBIND, 0, &ol_quantity);
dbbind(dbproc, 4, FLT8BIND, 0, &ol_amount);
dbbind(dbproc, 5, DATETIMEBIND, 0, &ol_delivery_d);

/* do for each row */
for (i=0; (code = dbnextrow(dbproc)) == REG_ROW && i<15; i++) {

/* move the information into the structure */
t->item[i].OL_SUPPLY_W_ID = ol_supply_w_id;
t->item[i].OL_I_ID = ol_i_id;
t->item[i].OL_QUANTITY = ol_quantity;
t->item[i].OL_AMOUNT = ol_amount;
from_sybase_date(ol_delivery_d, ol_delivery_d_DATE);
if (IsEmptyDate(&ol_delivery_d_DATE)) {
    t->item[i].OL_DELIVERY_DATE[0] = '0';
} else {
    fmt_date(t->item[i].OL_DELIVERY_DATE, &ol_delivery_d_DATE);
}
}

if (code != NO_MORE_ROWS) return NO;

/* remember how many rows we found */
t->ol_cnt = i;

if (dbresults(dbproc) != SUCCEED) return NO;

/* Code for TPC-C rev. 3.3 error checking. */
if (dbrows(dbproc) != SUCCEED) {
    t->status = E_INVALID_INPUT;
}

/* fetch the remaining information */
dbbind(dbproc, 1, INTBIND, 0, &t->C_ID);
dbbind(dbproc, 2, NTBSTRINGBIND, sizeof(t->C_LAST), t->C_LAST);
dbbind(dbproc, 3, NTBSTRINGBIND, sizeof(t->C_FIRST), t->C_FIRST);
dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(t->C_MIDDLE), t->C_MIDDLE);
dbbind(dbproc, 5, FLT8BIND, 0, &t->C_BALANCE);
dbbind(dbproc, 6, INTBIND, 0, &t->O_ID);
dbbind(dbproc, 7, DATETIMEBIND, 0, &o_entry_d);
dbbind(dbproc, 8, INTBIND, 0, &t->O_CARRIER_ID);
if (dbnextrow(dbproc) != REG_ROW) return NO;
if (dbcanquery(dbproc) != SUCCEED) return NO;

/* convert the date */
from_sybase_date(o_entry_d, o_entry_d_DATE);
fmt_date(t->O_ENTRY_DATE, &o_entry_d_DATE);

t->status = OK;
return YES;
}

delivery_transaction(t)
delivery_trans *t;
{
    ID d;

    int try;
    d = 1;

/* repeat until we give up trying */
for (try=0; try<MaxTries; try++)
    {

```

```

/* if the transaction succeeds, then done */
d = delivery_body(t, d);
if (d > 10) break;

/* clean up and try again */
dbcancel(dbproc);
sleep_before_retry();

/* don't retry if caused by operator error */
if (t->status == E_INVALID_INPUT) break;
}

/* any uncompleted districts have an error */
for ( ; d <= 10; d++)
    t->order[d-1].status = E_DB_ERROR;
}

int delivery_body(t, d)
delivery_trans *t;
ID d;
{
    dbrpcinit(dbproc, "delivery", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->W_ID));
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->O_CARRIER_ID));
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&d));
    if (dbrpcsend(dbproc) != SUCCEED) return d;
    if (dbsqllok(dbproc) != SUCCEED) return d;

    for ( ; d <= 10; d++)
    {
        /* Each order line is a separate query. Fetch the data */
        if (dbresults(dbproc) != SUCCEED) break;
        dbbind(dbproc, 1, INTBIND, 0, &t->order[d-1].O_ID);
        if (dbnextrow(dbproc) != REG_ROW) break;
        if (dbcanquery(dbproc) != SUCCEED) break;
        if (dbhasretstat(dbproc) && dbretstatus(dbproc) != 0) break;

        if (t->order[d-1].O_ID == 0) t->order[d-1].status = E_NOT_ENOUGH_ORDERS;
        else
            t->order[d-1].status = OK;
    }

    return d;
}

stocklev_transaction(t)
stocklev_trans *t;
{
    int try;

    /* repeat until we give up trying */
    for (try=0; try<MaxTries; try++)
    {
        /* if the transaction succeeds, then done */
        if (stocklev_body(t))
            break;

        /* clean up and try again */
        dbcancel(dbproc);
        sleep_before_retry();

        /* don't retry if caused by operator error */
        if (t->status == E_INVALID_INPUT) break;
    }

    /* if we finally gave up, then display a message */
    if (try >= MaxTries)
        t->status = E_DB_ERROR;
}

int stocklev_body(t)
stocklev_trans *t;
{
    int iid, uniq[500];
    int i, j, count;
    int duplicate_found;

    dbrpcinit(dbproc, "stock_level", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->W_ID));
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&t->D_ID));
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->threshold));
    if (dbrpcsend(dbproc) != SUCCEED) return NO;
    if (dbsqllok(dbproc) != SUCCEED) return NO;

    if (dbresults(dbproc) != SUCCEED) return NO;
    dbbind(dbproc, 1, INTBIND, 0, &iid);
    count = 0;
    while (dbnextrow(dbproc) == REG_ROW) {
        duplicate_found = 0;
        for (j = 0; j < count; j++) {
            if (iid == uniq[j]) {
                duplicate_found = 1;
                break;
            }
        }
        /* if this was a duplicate of something already found, then
           don't count it and continue */
        if (duplicate_found) continue;

        if (count < 500) {
            uniq[count++] = iid;
        } else {
            return NO;
        }
    }
    if (dbcanquery(dbproc) != SUCCEED) return NO;

    t->status = OK;
    t->low_stock = count;
    return YES;
}

int sleep_before_retry()
{
    delay(.1);
}

to_sybase_date(date, sybdate)
DATE *date;
DBDATETIME *sybdate;
{
    sybdate->dtday = date->day;
    sybdate->dttime = date->sec*300;
}

int deadlock = NO;

error_handler(dbproc, msgno, msgstate, severity, msgtext, srvname,
              procname, line)
/******
*****
error_handler deals with error messages
*****
*****/
DBPROCESS *dbproc;
DBINT msgno;
int msgstate;
int severity;
char *msgtext;
char *procname;
DBUSMALLINT line;
{
    if (deadlock)
        message("Error: Deadlock detected from %s line %d\n", procname, line);
    else
        message("Error #%d from %s line %d\n%s\n",
                msgno, procname, line, msgtext);
}

```

```

deadlock = NO;
return INT_CANCEL;
}

message_handler(dbproc, msgno, msgstate, severity, msgtext, srvname,
                procname, line)
/*****
message_handler deals with informational messages
*****/
DBPROCESS *dbproc;
DBINT msgno;
int msgstate;
int severity;
char *msgtext;
char *procname;
DBUSMALLINT line;
{
/* Ignore messages that will be passed to error handler anyway */
if (msgno == SYBESMSG)
return(SUCCESS);

/* Force an error for Deadlocks */
else if (msgno == 1205)
{
deadlock = YES;
message("Message: Deadlock detected from %s line %d\n", procname, line);
return(FAIL);
}

else if (msgno != 5701 && msgno != 5703 && msgno != 5704)
{
message("Message #%d from %s line %d\n%s\n",
        msgno, procname, line, msgtext);
return(FAIL);
}
}

```

---

## *client/service.c*

---

```

/*****
*****
@(#) Version: A.10.10 $Date: 97/12/15 10:53:26 $

(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****/

#include <unistd.h>
#include <sys/types.h>
#include "tpcc.h"
#include "atmi.h"

extern int userid;
char *cmd = NULL;

int tpsrvinit(argc, argv)
int argc;
char **argv;
{
char c;
int ret;

```

```

/*
* search for the options
* "-n" server number
* "-S" server program
* purpose: to get svr_id & proname for DVRY_LOG files
*/
while ((c = getopt(argc, argv, "n:S:h:")) != EOF) {
switch(c) {
case 'n':
userid = atoi(optarg);
break;
case 'S':
cmd = optarg;
break;
}
}

message("TUXEDO service %s has started\n", cmd);
ret = transaction_begin(userid);
results_open(userid);

return 0;
}

void NEWO_SVC(svcinfo)
TPSVCINFO *svcinfo;
{
neworder_transaction((neworder_trans *)svcinfo->data);
treturn(TPSUCCESS, 0, svcinfo->data, svcinfo->len, 0);
}

void PMT_SVC(svcinfo)
TPSVCINFO *svcinfo;
{
payment_transaction((payment_trans *)svcinfo->data);
treturn(TPSUCCESS, 0, svcinfo->data, svcinfo->len, 0);
}

void ORDS_SVC(svcinfo)
TPSVCINFO *svcinfo;
{
ordstat_transaction((ordstat_trans *)svcinfo->data);
treturn(TPSUCCESS, 0, svcinfo->data, svcinfo->len, 0);
}

void STKL_SVC(svcinfo)
TPSVCINFO *svcinfo;
{
stocklev_transaction((stocklev_trans *)svcinfo->data);
treturn(TPSUCCESS, 0, svcinfo->data, svcinfo->len, 0);
}

void DVRY_SVC(svcinfo)
TPSVCINFO *svcinfo;
{
delivery_trans *t = (delivery_trans *)svcinfo->data;
gettimeofday(t->deque, NULL);
delivery_transaction(t);
gettimeofday(t->complete, NULL);
results(t);

/* Why do we return things ? */
treturn(TPSUCCESS, 0, svcinfo->data, svcinfo->len, 0);
}

/*****
tpsrdone cleans up after the TPC transaction service
*****/
void tpsrdone()
{
transaction_done();
results_close();

/* Log a message saying we are done */
message("TUXEDO service %s has shutdown\n", cmd);

```

```
}
```

## A.4 TPC-C Stored Procedures

### *tpcc\_proc.sh*

```
#!/usr/bin/sh -f
#####
#####
#@(#) Version: A.10.10 $Date: 97/02/13 18:12:03 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#####
#####

# Stored procedure for TPC-C 3.0 on SQL Server 11.0 and later
# Copyright Sybase 1995

isql -Usa -P$PASSWORD <<EOF
use tpcc
go
if exists ( SELECT name FROM sysobjects WHERE name = 'neworder_local' )
    DROP PROC neworder_local
go

CREATE PROC neworder_local (
    @w_id          smallint,
    @d_id          tinyint,
    @c_id          int,
    @o_ol_cntint,

    @i_id   int = 0, @ol_qtytinyint = 0,
    @i_id2  int = 0, @ol_qty2tinyint = 0,
    @i_id3  int = 0, @ol_qty3tinyint = 0,
    @i_id4  int = 0, @ol_qty4tinyint = 0,
    @i_id5  int = 0, @ol_qty5tinyint = 0,
    @i_id6  int = 0, @ol_qty6tinyint = 0,
    @i_id7  int = 0, @ol_qty7tinyint = 0,
    @i_id8  int = 0, @ol_qty8tinyint = 0,
    @i_id9  int = 0, @ol_qty9tinyint = 0,
    @i_id10 int = 0, @ol_qty10tinyint = 0,
    @i_id11 int = 0, @ol_qty11tinyint = 0,
    @i_id12 int = 0, @ol_qty12tinyint = 0,
    @i_id13 int = 0, @ol_qty13tinyint = 0,
    @i_id14 int = 0, @ol_qty14tinyint = 0,
    @i_id15 int = 0, @ol_qty15tinyint = 0
)
as

declare
    @w_tax          real,          @d_tax          real,
    @c_last         char(16),    @c_creditchar(2),
    @c_discountreal,          @commit_flagint,

    @i_price float,
    @i_name         char(24),    @i_data         char(50),

    @s_quantitysmallint,
    @s_ytd          int,          @s_order_cntint,
    @s_dist         char(24),    @s_data         char(50),

    @ol_numberint,          @o_id          int,
    @o_entry_ddatetime,@b_g char(1),
    @B              char(1),    @G              char(1)
```

```
declare@ol_qty_smallint smallint, @minusonesmall smallint,
@tiny1 tinyint, @smallone smallint,
@small0 smallint, @small91 smallint, @small10 smallint,
@d_id_int int, @original char(10)
```

```
declare c_no_wdc CURSOR FOR
SELECT w_tax, d_tax, d_next_o_id,
       c_last, c_discount, c_credit, getdate()
FROM   district HOLDLOCK,
       warehouse HOLDLOCK,
       customer (index c_clu prefetch 2 lru) HOLDLOCK
WHERE  d_w_id = @w_id
AND    d_id   = @d_id
AND    w_id   = @w_id
AND    c_w_id = @w_id
AND    c_d_id = @d_id
AND    c_id   = @c_id
FOR UPDATE OF d_next_o_id
```

```
BEGIN TRANSACTION NO
select @tiny1 = 1, @smallone = 1, @minusonesmall = -1,
       @small0 = 0, @small91 = 91, @small10 = 10,
       @original = "%ORIGINAL%", @B = "B", @G = "G",
       @d_id_int = @d_id, @commit_flag = 1, @ol_number = 0
```

```
OPEN c_no_wdc
FETCH c_no_wdc INTO
    @w_tax, @d_tax, @o_id,
    @c_last, @c_discount, @c_credit,
    @o_entry_d
UPDATE district
    SET d_next_o_id = @o_id + 1
    WHERE CURRENT OF c_no_wdc
CLOSE c_no_wdc
```

```
while (@ol_number < @o_ol_cnt) begin
    SELECT @ol_number = @ol_number + 1,
           @i_id = case @ol_number
```

```
    when 1 then @i_id2
    when 2 then @i_id3
    when 3 then @i_id4
    when 4 then @i_id5
    when 5 then @i_id6
    when 6 then @i_id7
    when 7 then @i_id8
    when 8 then @i_id9
    when 9 then @i_id10
    when 10 then @i_id11
    when 11 then @i_id12
    when 12 then @i_id13
    when 13 then @i_id14
    when 14 then @i_id15
    else @i_id
    end
```

```
, @ol_qty = case @ol_number
    when 1 then @ol_qty2
    when 2 then @ol_qty3
    when 3 then @ol_qty4
    when 4 then @ol_qty5
    when 5 then @ol_qty6
    when 6 then @ol_qty7
    when 7 then @ol_qty8
    when 8 then @ol_qty9
    when 9 then @ol_qty10
    when 10 then @ol_qty11
    when 11 then @ol_qty12
    when 12 then @ol_qty13
    when 13 then @ol_qty14
    when 14 then @ol_qty15
    else @ol_qty
    end
```

```
/* set i_id, ol_qty for this lineitem */
/* this is replaced by case statement */
```

```
/* convert c_no_is cursor to a simple select */
/* get item data (no one update item) */
```

```

select @i_price = i_price,
       @i_name = i_name,
       @i_data = i_data
from item HOLDLOCK
where i_id = @i_id

if (@@rowcount = 0)
begin
select @commit_flag = 0
select NULL, NULL, NULL, NULL
continue
end
/*Otherwise if the item is found */

/* s_ytd == int, ol_qty == tiny, s_quantity == small,
91 = int, 0 = int */

update stock
set s_ytd = s_ytd + @ol_qty,
    @s_quantity = s_quantity - @ol_qty +
    case when (s_quantity - @ol_qty < @small10)
    then @small91 else @small0 end,
    s_quantity = s_quantity - @ol_qty +
    case when (s_quantity - @ol_qty < @small10)
    then @small91 else @small0 end,
    s_order_cnt = s_order_cnt + @smallone,
    @s_data = s_data,
    @s_dist = case @d_id_int
    when 1 then s_dist_01
    when 2 then s_dist_02
    when 3 then s_dist_03
    when 4 then s_dist_04
    when 5 then s_dist_05
    when 6 then s_dist_06
    when 7 then s_dist_07
    when 8 then s_dist_08
    when 9 then s_dist_09
    when 10 then s_dist_10
    end
    where s_w_id = @w_id and
    s_i_id = @i_id
if (@@rowcount = 0)
begin
select @commit_flag = 0
select NULL, NULL, NULL, NULL
continue
end
/*Otherwise if the Stock is found */

select @ol_qty_smallint = @ol_qty
INSERT INTO order_line (
    ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
    ol_supply_w_id, ol_delivery_d, ol_quantity,
    ol_amount, ol_dist_info)
VALUES (
    @o_id, @d_id, @w_id, @ol_number, @i_id,
    @w_id, "19000101", @ol_qty_smallint,
    @ol_qty * @i_price, @s_dist)
/* send line-item data to client */
select
    @i_name,
    @i_price,
    @s_quantity,
    b_g = case when ((patindex(@original, @i_data) > 0) and
    (patindex(@original, @s_data) > 0))
    then @B else @G end
end /* while */

INSERT INTO orders (
    o_id, o_c_id, o_d_id, o_w_id,
    o_entry_d, o_carrier_id, o_ol_cnt, o_all_local)
VALUES (
    @o_id, @c_id, @d_id, @w_id,
    @o_entry_d, @minusonesmall, @o_ol_cnt, @tiny1)
INSERT INTO new_order (no_o_id, no_d_id, no_w_id)
VALUES (@o_id, @d_id, @w_id)

if (@commit_flag = 1)
        commit transaction NO
    else
        rollback transaction NO

select
        /* Return to client */
        @w_tax, @d_tax, @o_id, @c_last,
        @c_discount, @c_credit, @o_entry_d
go

if exists ( SELECT name FROM sysobjects WHERE name = 'neworder_remote' )
DROP PROC neworder_remote
go
CREATE PROC neworder_remote (
    @w_id          smallint,
    @d_id          tinyint,
    @c_id          int,
    @o_ol_cntint,

    @i_id  int=0, @s_w_idsmallint=0, @ol_qtytinyint=0,
    @i_id2 int=0, @s_w_id2smallint=0, @ol_qty2tinyint=0,
    @i_id3 int=0, @s_w_id3smallint=0, @ol_qty3tinyint=0,
    @i_id4 int=0, @s_w_id4smallint=0, @ol_qty4tinyint=0,
    @i_id5 int=0, @s_w_id5smallint=0, @ol_qty5tinyint=0,
    @i_id6 int=0, @s_w_id6smallint=0, @ol_qty6tinyint=0,
    @i_id7 int=0, @s_w_id7smallint=0, @ol_qty7tinyint=0,
    @i_id8 int=0, @s_w_id8smallint=0, @ol_qty8tinyint=0,
    @i_id9 int=0, @s_w_id9smallint=0, @ol_qty9tinyint=0,
    @i_id10 int=0, @s_w_id10 smallint=0, @ol_qty10tinyint=0,
    @i_id11 int=0, @s_w_id11 smallint=0, @ol_qty11tinyint=0,
    @i_id12 int=0, @s_w_id12 smallint=0, @ol_qty12tinyint=0,
    @i_id13 int=0, @s_w_id13 smallint=0, @ol_qty13tinyint=0,
    @i_id14 int=0, @s_w_id14 smallint=0, @ol_qty14tinyint=0,
    @i_id15 int=0, @s_w_id15 smallint=0, @ol_qty15tinyint=0
)
as
declare
    @w_tax          real,          @d_tax          real,
    @c_last        char(16),      @c_creditchar(2),
    @c_discountreal,             @commit_flagint,

    @i_price float,
    @i_name        char(24),      @i_data        char(50),

    @s_quantitysmallint,
    @s_ytd         int,          @s_order_cntint,
    @s_dist        char(24),      @s_data        char(50),
    @s_remote_cntint,             @remote        int,

    @ol_numbertinyint, @o_id          int,
    @o_entry_ddatetime, @b_g          char(1)

declare @ol_qty_smallint smallint, @minusonesmall smallint,
    @tiny0 tinyint, @smallone smallint,
    @small0 smallint, @small91 smallint, @small10 smallint,
    @B char(1), @G char(1),
    @d_id_int int, @original char(10)

declare c_no_wdc CURSOR FOR
SELECT w_tax, d_tax, d_next_o_id,
    c_last, c_discount, c_credit, getdate()
FROM district HOLDLOCK,
    warehouse HOLDLOCK,
    customer (index c_clu prefetch 2 lru) HOLDLOCK
WHERE d_w_id = @w_id
AND d_id = @d_id
AND w_id = d_w_id
AND c_w_id = w_id
AND c_d_id = d_id
AND c_id = @c_id
FOR UPDATE OF d_next_o_id

BEGIN TRANSACTION NOREM
select @tiny0 = 0, @smallone = 1, @minusonesmall = -1,
    @small0 = 0, @small91 = 91, @small10 = 10,
    @original = "%ORIGINAL%", @B = "B", @G = "G",
    @d_id_int = @d_id, @commit_flag = 1, @ol_number = 0

OPEN c_no_wdc

```

```

FETCH c_no_wdc INTO
    @w_tax, @d_tax, @o_id,
    @c_last, @c_discount, @c_credit, @o_entry_d
UPDATE district
    SET d_next_o_id = @o_id + 1
    WHERE CURRENT OF c_no_wdc
CLOSE c_no_wdc

```

```

while (@ol_number < @o_ol_cnt) begin
    SELECT @ol_number = @ol_number + 1,
    @i_id = case @ol_number
        when 1 then @i_id2
        when 2 then @i_id3
        when 3 then @i_id4
        when 4 then @i_id5
        when 5 then @i_id6
        when 6 then @i_id7
        when 7 then @i_id8
        when 8 then @i_id9
        when 9 then @i_id10
        when 10 then @i_id11
        when 11 then @i_id12
        when 12 then @i_id13
        when 13 then @i_id14
        when 14 then @i_id15
        else @i_id
    end
, @ol_qty = case @ol_number
    when 1 then @ol_qty2
    when 2 then @ol_qty3
    when 3 then @ol_qty4
    when 4 then @ol_qty5
    when 5 then @ol_qty6
    when 6 then @ol_qty7
    when 7 then @ol_qty8
    when 8 then @ol_qty9
    when 9 then @ol_qty10
    when 10 then @ol_qty11
    when 11 then @ol_qty12
    when 12 then @ol_qty13
    when 13 then @ol_qty14
    when 14 then @ol_qty15
    else @ol_qty
end
, @s_w_id = case @ol_number
    when 1 then @s_w_id2
    when 2 then @s_w_id3
    when 3 then @s_w_id4
    when 4 then @s_w_id5
    when 5 then @s_w_id6
    when 6 then @s_w_id7
    when 7 then @s_w_id8
    when 8 then @s_w_id9
    when 9 then @s_w_id10
    when 10 then @s_w_id11
    when 11 then @s_w_id12
    when 12 then @s_w_id13
    when 13 then @s_w_id14
    when 14 then @s_w_id15
    else @s_w_id
end

```

/\* convert c\_no\_is cursor to a simple select \*/

/\* get item data (no one update item) \*/

```

select @i_price = i_price,
    @i_name = i_name,
    @i_data = i_data
from item HOLDLOCK
where i_id = @i_id

```

```

if (@@rowcount = 0)
begin
    select @commit_flag = 0
    select NULL, NULL, NULL, NULL
    continue
end

```

/\* Otherwise if the item is found \*/  
update stock

```

set s_ytd = s_ytd + @ol_qty,
@s_quantity = s_quantity - @ol_qty +
    case when (s_quantity - @ol_qty < @small10)
    then @small91 else @small10 end,
s_quantity = s_quantity - @ol_qty +
    case when (s_quantity - @ol_qty < @small10)
    then @small91 else @small10 end,
@s_data = s_data,
@s_dist = case @d_id_int
    when 1 then s_dist_01
    when 2 then s_dist_02
    when 3 then s_dist_03
    when 4 then s_dist_04
    when 5 then s_dist_05
    when 6 then s_dist_06
    when 7 then s_dist_07
    when 8 then s_dist_08
    when 9 then s_dist_09
    when 10 then s_dist_10
end,
s_order_cnt = s_order_cnt + @smallone,
s_remote_cnt = s_remote_cnt +
case when (@s_w_id = @w_id)
then 0 else 1 end
where s_w_id = @w_id and
s_i_id = @i_id

```

```

if (@@rowcount = 0)
begin
    select @commit_flag = 0
    select NULL, NULL, NULL, NULL
    continue
end

```

```

select @ol_qty_smallint = @ol_qty
INSERT INTO order_line (
    ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
    ol_supply_w_id, ol_delivery_d, ol_quantity,
    ol_amount, ol_dist_info)
VALUES (
    @o_id, @d_id, @w_id, @ol_number, @i_id,
    @s_w_id, "19000101", @ol_qty_smallint,
    @ol_qty * @i_price, @s_dist)

```

select @i\_name, /\* Return to client \*/

```

    @i_price,
    @s_quantity,
    b_g = case when ((patindex(@original, @i_data) > 0) and
        (patindex(@original, @s_data) > 0))
    then @B else @G end
end

```

```

INSERT INTO orders (
    o_id, o_c_id, o_d_id, o_w_id,
    o_entry_d, o_carrier_id, o_ol_cnt, o_all_local)
VALUES (
    @o_id, @c_id, @d_id, @w_id,
    @o_entry_d, @minusonesmall, @o_ol_cnt, @tiny0)
INSERT INTO new_order (no_o_id, no_d_id, no_w_id)
VALUES (@o_id, @d_id, @w_id)

```

```

if (@commit_flag = 1)
    commit transaction NOREM
else
    rollback transaction NOREM

```

```

select /* Return to client */
    @w_tax, @d_tax, @o_id, @c_last,
    @c_discount, @c_credit, @o_entry_d

```

```

go
if exists (select * from sysobjects where name = 'payment_byid')
DROP PROC payment_byid

```

go  
/\* NOTE: \_numtoft8 coming from converting output of "c\_credit\_lim"  
to a float 8. We need to store c\_credit\_lim as a "float"  
instead of a numeric(12), should also store c\_discount as  
a "float" \*/

CREATE PROC payment\_byid

```

        @w_id          smallint,  @c_w_id          smallint,
        @h_amount float,
        @d_id          tinyint,   @c_d_id          tinyint,
        @c_id          int
as
declare @c_last      char(16)

declare @w_street_1 char(20), @w_street_2 char(20),
        @w_city      char(20),  @w_state char(2),
        @w_zip       char(9),   @w_name     char(10),
        @w_ytd       float

declare @d_street_1 char(20), @d_street_2 char(20),
        @d_city      char(20),  @d_state char(2),
        @d_zip       char(9),   @d_name     char(10),
        @d_ytd       float

declare @c_first char(16), @c_middle char(2),
        @c_street_1 char(20), @c_street_2 char(20),
        @c_city      char(20),  @c_state char(2),
        @c_zip       char(9),   @c_phone char(16),
        @c_since datetime, @c_credit char(2),
        @c_credit_lim numeric(12,0), @c_balance float,
        @c_discount real,
        @i          smallint,
        @data1      char(250), @data2      char(250),
        @c_data_1 char(250), @c_data_2 char(250)

declare @screen_data char(200), @today datetime

declare @w_id_new smallint, @small_one smallint, @BC char(2),
        @onehundred float, @space char(4)

declare c_pay_wd CURSOR FOR
SELECT w_id, w_street_1, w_street_2, w_city,
       w_state, w_zip, w_name, w_ytd,
       d_street_1, d_street_2, d_city,
       d_state, d_zip, d_name, d_ytd
FROM   district HOLDLOCK,
       warehouse HOLDLOCK
WHERE  d_w_id = @w_id
AND    d_id = @d_id
AND    w_id = d_w_id
FOR UPDATE OF w_ytd, d_ytd

BEGIN TRANSACTION PID
select @small_one = 1, @BC = "BC", @onehundred = 100.0, @space = " "

OPEN c_pay_wd
FETCH c_pay_wd INTO
        @w_id_new, @w_street_1, @w_street_2, @w_city,
        @w_state, @w_zip, @w_name, @w_ytd,
        @d_street_1, @d_street_2, @d_city,
        @d_state, @d_zip, @d_name, @d_ytd
UPDATE district
SET    d_ytd = @d_ytd + @h_amount
WHERE CURRENT OF c_pay_wd
UPDATE warehouse
SET    w_ytd = @w_ytd + @h_amount
WHERE CURRENT OF c_pay_wd
CLOSE c_pay_wd

/* Customer data */
UPDATE customer SET
        @c_first = c_first
        , @c_middle = c_middle
        , @c_last = c_last
        , @c_street_1 = c_street_1
        , @c_street_2 = c_street_2
        , @c_city = c_city
        , @c_state = c_state
        , @c_zip = c_zip
        , @c_phone = c_phone
        , @c_credit = c_credit
        , @c_credit_lim = c_credit_lim
        , @c_discount = c_discount
        , c_balance = c_balance - @h_amount
        , @c_balance = c_balance - @h_amount
        , c_ytd_payment = c_ytd_payment + @h_amount
        , c_payment_cnt = c_payment_cnt + @small_one
        , @c_since = c_since
        , @data1 = c_data1
        , @data2 = c_data2
        , @today = getdate()
where
        c_id = @c_id
        and c_w_id = @c_w_id
        and c_d_id = @c_d_id

if (@c_credit = @BC)
begin
SELECT @c_data_2 =
        substring(@data1, 209, 42) +
        substring(@data2, 1, 208)
        , @c_data_1 =
        convert(char(5), @c_id) +
        convert(char(4), @c_d_id) +
        convert(char(5), @c_w_id) +
        convert(char(4), @d_id) +
        convert(char(5), @w_id) +
        convert(char(19), @h_amount/@onehundred) + substring(@data1, 1, 208)

UPDATE customer SET
        c_data1 = @c_data_1
        , c_data2 = @c_data_2
        , @screen_data = substring(@c_data_1, 1, 200)
WHERE
        c_id = @c_id
        AND c_w_id = @c_w_id
        AND c_d_id = @c_d_id
end /* if */

/* Create the history record */
INSERT INTO history (
        h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
        h_date, h_amount, h_data)
VALUES (
        @c_id, @c_d_id, @c_w_id, @d_id, @w_id_new,
        @today, @h_amount, (@w_name + @space + @d_name))

COMMIT TRANSACTION PID

select /* Return to client */
        @c_id,
        @c_last,
        @today,
        @w_street_1,
        @w_street_2,
        @w_city,
        @w_state,
        @w_zip,
        @d_street_1,
        @d_street_2,
        @d_city,
        @d_state,
        @d_zip,
        @c_first,
        @c_middle,
        @c_street_1,
        @c_street_2,
        @c_city,
        @c_state,
        @c_zip,
        @c_phone,
        @c_since,
        @c_credit,
        @c_credit_lim,
        @c_discount,
        @c_balance,
        @screen_data
go
if exists (select * from sysobjects where name = 'payment_byname')
DROP PROC payment_byname
go
CREATE PROC payment_byname
        @w_id          smallint,  @c_w_id          smallint,

```

```

    @h_amount float,
    @d_id        tinyint,    @c_d_id        tinyint,
    @c_last      char(16)
as
declare @n      int,        @c_id        int

declare @w_street_1 char(20), @w_street_2 char(20),
        @w_city      char(20), @w_state char(2),
        @w_zip       char(9),  @w_name      char(10),
        @w_ytd       float

declare @d_street_1 char(20), @d_street_2 char(20),
        @d_city      char(20), @d_state char(2),
        @d_zip       char(9),  @d_name      char(10),
        @d_ytd       float

declare @c_first char(16), @c_middle char(2),
        @c_street_1 char(20), @c_street_2 char(20),
        @c_city      char(20), @c_state char(2),
        @c_zip       char(9),  @c_phone char(16),
        @c_since datetime, @c_credit char(2),
        @c_credit_lim numeric(12,0), @c_balance float,
        @c_discount real,
        @l          smallint,
        @data1      char(250), @data2      char(250),
        @c_data_1 char(250), @c_data_2 char(250)

declare @screen_data char(200), @today datetime

declare @w_id_new smallint, @small_one smallint, @BC char(2),
        @onehundred float, @space char(4)

declare c_pay_wd CURSOR FOR
    SELECT w_id, w_street_1, w_street_2, w_city,
           w_state, w_zip, w_name, w_ytd,
           d_street_1, d_street_2, d_city,
           d_state, d_zip, d_name, d_ytd
    FROM   district HOLDLOCK,
           warehouse HOLDLOCK
    WHERE  d_w_id = @w_id
    AND    d_id = @d_id
    AND    w_id = d_w_id
    FOR UPDATE OF w_ytd, d_ytd

declare c_find CURSOR FOR
    SELECT c_id
    FROM   customer (index c_non1 prefetch 2 lru) HOLDLOCK
    WHERE  c_w_id = @c_w_id
    AND    c_d_id = @c_d_id
    AND    c_last = @c_last
    ORDER BY c_w_id, c_d_id, c_last, c_first, c_id
    FOR READ ONLY

BEGIN TRANSACTION PNM
select @small_one = 1, @BC = "BC", @onehundred = 100.0, @space = " "
SELECT @n = (count(*)+1)/2
FROM   customer (index c_non1 prefetch 2 lru) HOLDLOCK
WHERE  c_w_id = @c_w_id and
       c_d_id = @c_d_id and
       c_last = @c_last

OPEN c_find
while (@n>0) begin
    FETCH c_find INTO @c_id
    SELECT @n = @n-1
end
CLOSE c_find

OPEN c_pay_wd
FETCH c_pay_wd INTO
    @w_id_new, @w_street_1, @w_street_2, @w_city,
    @w_state, @w_zip, @w_name, @w_ytd,
    @d_street_1, @d_street_2, @d_city,
    @d_state, @d_zip, @d_name, @d_ytd
UPDATE district
SET   d_ytd = @d_ytd + @h_amount
WHERE CURRENT OF c_pay_wd
UPDATE warehouse
SET   w_ytd = @w_ytd + @h_amount
WHERE CURRENT OF c_pay_wd

```

```

CLOSE c_pay_wd
/* Customer data */
UPDATE customer SET
    @c_first = c_first
    , @c_middle = c_middle
    , @c_last = c_last
    , @c_street_1 = c_street_1
    , @c_street_2 = c_street_2
    , @c_city = c_city
    , @c_state = c_state
    , @c_zip = c_zip
    , @c_phone = c_phone
    , @c_credit = c_credit
    , @c_credit_lim = c_credit_lim
    , @c_discount = c_discount
    , c_balance = c_balance - @h_amount
    , @c_balance = c_balance - @h_amount
    , c_ytd_payment = c_ytd_payment + @h_amount
    , c_payment_cnt = c_payment_cnt + @small_one
    , @c_since = c_since
    , @data1 = c_data1
    , @data2 = c_data2
    , @today = getdate()
where
    c_id = @c_id
    and c_w_id = @c_w_id
    and c_d_id = @c_d_id

SELECT @screen_data = NULL
if (@c_credit = @BC)
begin
    SELECT @c_data_2 =
        substring(@data1, 209, 42) +
        substring(@data2, 1, 208)
    , @c_data_1 =
        convert(char(5), @c_id) +
        convert(char(4), @c_d_id) +
        convert(char(5), @c_w_id) +
        convert(char(4), @d_id) +
        convert(char(5), @w_id) +
        convert(char(19), @h_amount/@onehundred) + substring(@data1, 1, 208)

    UPDATE customer SET
        c_data1 = @c_data_1
        , c_data2 = @c_data_2
        , @screen_data = substring(@c_data_1, 1, 200)
    WHERE
        c_id = @c_id
        AND c_w_id = @c_w_id
        AND c_d_id = @c_d_id
end /* if */

INSERT INTO history (
    h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
    h_date, h_amount, h_data)
VALUES (
    @c_id, @c_d_id, @c_w_id, @d_id, @w_id_new,
    @today, @h_amount, (@w_name + @space + @d_name))

COMMIT TRANSACTION PNM

select /* Return to client */
    @c_id,
    @c_last,
    @today,
    @w_street_1,
    @w_street_2,
    @w_city,
    @w_state,
    @w_zip,
    @d_street_1,
    @d_street_2,
    @d_city,
    @d_state,
    @d_zip,
    @c_first,

```



```

@c_middle,
@c_street_1,
@c_street_2,
@c_city,
@c_state,
@c_zip,
@c_phone,
@c_since,
@c_credit,
@c_credit_lim,
@c_discount,
@c_balance,
@screen_data

go
if exists (select * from sysobjects where name = 'order_status_byid')
  DROP PROC order_status_byid
go
CREATE PROC order_status_byid
  @w_id          smallint,
  @d_id          tinyint,
  @c_id          int
as

DECLARE@o_id      int,
@o_entry_ddatetime,
@o_carrier_idsmallint

BEGIN TRANSACTION OSID

/* Get the latest order made by the customer */
SELECT @o_id = o_id, @o_carrier_id = o_carrier_id,
       @o_entry_d = o_entry_d
FROM   orders (index o_clu prefetch 16 lru) HOLDLOCK
WHERE  o_w_id = @w_id
AND    o_d_id = @d_id
AND    o_c_id = @c_id
/* ORDER BY o_w_id, o_d_id, o_id */

/* Select order lines for the current order */
select /* Return multiple rows to client */
       ol_supply_w_id,
       ol_i_id,
       ol_quantity,
       ol_amount,
       ol_delivery_d
FROM   order_line HOLDLOCK
WHERE  ol_o_id = @o_id
AND    ol_d_id = @d_id
AND    ol_w_id = @w_id

select /* Return single row to client */
       @c_id, c_last, c_first, c_middle, c_balance,
       @o_id,
       @o_entry_d,
       @o_carrier_id
FROM   customer (index c_clu prefetch 2 lru) HOLDLOCK
WHERE  c_id = @c_id
AND    c_d_id = @d_id
AND    c_w_id = @w_id

COMMIT TRANSACTION OSID
go
if exists (select * from sysobjects where name = 'order_status_byname')
  DROP PROC order_status_byname
go
CREATE PROC order_status_byname
  @w_id          smallint,
  @d_id          tinyint,
  @c_last        char(16)
as

DECLARE@o_id      int,
@o_entry_ddatetime,
@o_carrier_idsmallint

declare@n          int, @c_id      int
declare c_find CURSOR FOR
  SELECT c_id
  FROM   customer (index c_non1 prefetch 2 lru) HOLDLOCK

WHERE  c_w_id = @w_id
AND    c_d_id = @d_id
AND    c_last = @c_last
ORDER BY c_w_id, c_d_id, c_last, c_first, c_id
FOR READ ONLY

BEGIN TRANSACTION OSNM
SELECT @n = (count(*)+1)/2
FROM   customer (index c_non1 prefetch 2 lru) HOLDLOCK
WHERE  c_w_id = @w_id and
       c_d_id = @d_id and
       c_last = @c_last

OPEN c_find
while (@n>0) begin
  FETCH c_find INTO @c_id
  SELECT @n = @n-1
end
CLOSE c_find

/* Get the latest order made by the customer */
SELECT @o_id = o_id, @o_carrier_id = o_carrier_id,
       @o_entry_d = o_entry_d
FROM   orders (index o_clu prefetch 16 lru) HOLDLOCK
WHERE  o_w_id = @w_id
AND    o_d_id = @d_id
AND    o_c_id = @c_id
/* ORDER BY o_w_id, o_d_id, o_id */

/* Select order lines for the current order */
select /* Return multiple rows to client */
       ol_supply_w_id,
       ol_i_id,
       ol_quantity,
       ol_amount,
       ol_delivery_d
FROM   order_line HOLDLOCK
WHERE  ol_o_id = @o_id
AND    ol_d_id = @d_id
AND    ol_w_id = @w_id

select /* Return single row to client */
       @c_id, c_last, c_first, c_middle, c_balance,
       @o_id,
       @o_entry_d,
       @o_carrier_id
FROM   customer (index c_clu prefetch 2 lru) HOLDLOCK
WHERE  c_id = @c_id
AND    c_d_id = @d_id
AND    c_w_id = @w_id

COMMIT TRANSACTION OSNM
go
if exists (select * from sysobjects where name = 'delivery')
  drop proc delivery
go
CREATE PROC delivery
  @w_id          smallint,
  @o_carrier_idsmallint,
  @d_id          tinyint
as

declare@no_o_id int, @o_c_id      int,
@ol_total float, @ten          tinyint,
@one_tiny      tinyint, @one_small smallint

declare c_del_no CURSOR FOR
  SELECT no_o_id
  FROM   new_order (index no_clu) HOLDLOCK
  WHERE  no_d_id = @d_id
  AND    no_w_id = @w_id
  FOR UPDATE
  /*
  ** The only purpose of the index hint in the above is to ensure
  ** that the clustered index is used. As it turns out, our optimizer
  ** chooses the clustered index anyway -- with or without the hint.
  */
begin
  select @ten = 10, @one_small = 1, @one_tiny = 1

```

```

while (@d_id <= @ten) begin

    BEGIN TRANSACTION DEL
    OPEN c_del_no
    FETCH c_del_no INTO @no_o_id

    if (@@sqlstatus != 0)
    begin
        COMMIT TRANSACTION DEL
        select NULL
        CLOSE c_del_no
        select @d_id = @d_id + @one_tiny
        continue
    end

    DELETE FROM new_order
    WHERE CURRENT OF c_del_no
    CLOSE c_del_no

    /* Using the 'update' enhancement */

    UPDATE orders
    SET o_carrier_id = @o_carrier_id,
        @o_c_id = o_c_id,
        @ol_total = 0.0
    WHERE o_id = @no_o_id
    AND o_d_id = @d_id
    AND o_w_id = @w_id

    UPDATE order_line
    SET ol_delivery_d = getdate(),
        @ol_total = ol_amount + @ol_total
    WHERE ol_o_id = @no_o_id
    AND ol_d_id = @d_id
    AND ol_w_id = @w_id
    UPDATE customer
    SET c_balance = c_balance + @ol_total,
        c_delivery_cnt = c_delivery_cnt + @one_small
    WHERE c_id = @o_c_id
    AND c_d_id = @d_id
    AND c_w_id = @w_id

    COMMIT TRANSACTION DEL

    select /* Return to client */
        @no_o_id
    select @d_id = @d_id + @one_tiny
    end /* while @d_id... */
end
go
if exists ( SELECT name FROM sysobjects WHERE name = 'stock_level' )
DROP PROC stock_level
go

CREATE PROC stock_level
    @w_id smallint,
    @d_id tinyint,
    @threshold smallint
as
    select s_i_id
    FROM district,
        order_line (index ol_clu prefetch 2 lru),
        stock (index s_clu prefetch 2 lru)
    WHERE d_w_id = @w_id
    AND d_id = @d_id
    AND ol_w_id = @w_id
    AND ol_d_id = @d_id
    AND ol_o_id between (d_next_o_id - 20) and (d_next_o_id - 1)
    AND s_w_id = ol_w_id
    AND s_i_id = ol_i_id
    AND s_quantity < @threshold
go
EOF

```

# Appendix B: Database Design

---

The source code for the process to define, create and populate the Sybase Adaptive Server Enterprise 11.5 TPC-C database is included in this appendix.

---

## B.1 Main Shell Scripts

---

### *build*

---

---

```
#!/usr/bin/csh
##*****
#@(#) Version: A.10.10 $Date: 97/12/15 10:49:50 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
##*****

source ~tpcc/TESTENV
build.real
```

---

---

### *build.real*

---

---

```
#!/usr/bin/ksh
##*****
#@(#) Version: A.10.10 $Date: 97/12/15 10:49:50 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
##*****

remsh $SERVER -n create_devices
load_database
```

---

---

## *create\_devices*

---

---

```
#!/usr/bin/sh -f
#*****
#*****
#@(#) Version: A.10.10 $Date: 97/12/15 13:17:08 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*****
#*****

#
# Clean up some stuff first
#
rm -rf ~tpcc/logs/dev_create.OUT
rm -rf ~tpcc/logs/*.log

#
# Create the log file
#
exec > ~tpcc/logs/dev_create.OUT 2>&1

shutdown_server.sh
rm -f ~tpcc/dev/errorlog

#define SQL_RELEASE, MASTER_DEVICE etc
echo `date` "Started bld_system"
bmbinary=$SQL_RELEASE/bin/buildmaster

# Build the device.
(cd ~tpcc/dev; `devcreate.sh buildmaster $bmbinary < ~tpcc/load/devices`)

# Boot server, run installmaster, shutdown server

run_server - -T1608
isql -Usa -P < $SQL_RELEASE/scripts/installmaster > ~tpcc/logs/$$_im.log

# Reboot, build devices, database, and segments
echo `date` "Creating devices, databases and segments"
devcreate.sh sql System10 < ~tpcc/load/devices | isql -e -Usa -P
echo `date` " Finished building database"

# Create tables, some indexes, and administrative procs.
tpcc_tables.sh

# Truncate log, checkpoint, and shutdown
dumptran_server.sh master
dumptran_server.sh tpcc
shutdown_server.sh

# startup server without logging to prepare for build

run_server - -T699
```

---

---

## *load\_database*

---

---

```
#!/usr/bin/ksh
#*****
#*****
#@(#) Version: A.10.10 $Date: 97/12/15 13:11:36 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*****
#*****

rm -rf ~tpcc/logs/build.OUT
exec > ~tpcc/logs/build.OUT 2>&1
```

```
#
# Load the data; shutdown again.
#
echo `date` " Started loading data"

isql -Usa -P$PASSWORD << EOF
sp_dboption tpcc,"select into/bulkcopy",true
go
EOF

# load small tables in serial
load_wdi
# load large tables in parallel
load_stock &
sleep 5
load_customer &
sleep 5
load_orderline &
sleep 5
load_orders &
sleep 5
load_history &
sleep 5
load_neworder &

wait

tpcc_proc.sh

echo `date` -- Done building, get the table sizes
table_size.sh

shutdown_server.sh

echo `date` "Finished bld_system"
```

---

---

## *load\_wdi*

---

---

```
#!/usr/bin/sh -f

load -t item
load -t warehouse 1 $DB_SIZE
load -t district 1 $DB_SIZE

isql -Usa -P$PASSWORD << EOF
/* This script will create the TPC-C indexes that are best
   created after the load. */
use tpcc
go

create unique clustered index w_clu
  on warehouse(w_id)
  with sorted_data
  on Swarehouse

go
dbcc tune(indextrips, 100, warehouse)
go

create unique clustered index d_clu
  on district(d_w_id, d_id)
  with sorted_data
  on Sdistrict

go
dbcc tune(indextrips, 100, district)
go

create unique clustered index i_clu
  on item(i_id)
  with sorted_data
  on Sitem
```

```
go
dbcc tune(indextrips, 10, item)
go
EOF
```

---

---

## *load\_stock*

---

---

```
#!/usr/bin/sh -f

isql -Usa -P$PASSWORD << EOF
use tpcc
go
dbcc iosize("tpcc", "stock", 16)
go
EOF
```

```
load -t stock 1 $DB_SIZE
```

```
isql -Usa -P$PASSWORD << EOF
use tpcc
go
drop index stock.s_clu
go
create unique clustered index s_clu
    on stock(s_i_id, s_w_id)
    with sorted_data
    on Sstock
go
dbcc tune(indextrips, 10, stock)
go
EOF
```

---

---

## *load\_customer*

---

---

```
#!/usr/bin/sh -f

isql -Usa -P$PASSWORD << EOF
use tpcc
go
dbcc iosize("tpcc", "customer", 16)
go
EOF
```

```
load -t customer 1 $DB_SIZE
```

```
isql -Usa -P$PASSWORD << EOF
use tpcc
go
```

```
drop index customer.c_clu
go
create unique clustered index c_clu
    on customer(c_w_id, c_i_id, c_d_id)
    with sorted_data
    on Scustomer
go
```

```
create unique nonclustered index c_non1
    on customer(c_w_id, c_d_id, c_last, c_first, c_id)
    with fillfactor = 100
    on Sc_index
go
```

```
EOF
```

---

---

## *load\_orderline*

---

---

```
#!/usr/bin/sh -f
```

```
isql -Usa -P$PASSWORD << EOF
use tpcc
go
dbcc iosize("tpcc", "order_line", 16)
go
EOF
```

```
load -t orderline 1 $DB_SIZE
```

---

---

## *load\_orders*

---

---

```
#!/usr/bin/sh -f
```

```
isql -Usa -P$PASSWORD << EOF
use tpcc
go
dbcc iosize("tpcc", "orders", 16)
go
EOF
```

```
load -t orders 1 $DB_SIZE
```

---

---

## *load\_history*

---

---

```
#!/usr/bin/sh -f
```

```
isql -Usa -P$PASSWORD << EOF
use tpcc
go
dbcc iosize("tpcc", "history", 16)
go
EOF
```

```
load -t history 1 $DB_SIZE
```

---

---

## *load\_neworder*

---

---

```
#!/usr/bin/sh -f
```

```
isql -Usa -P$PASSWORD << EOF
use tpcc
go
```

```
dbcc iosize("tpcc", "new_order", 16)
go
EOF
```

```
load -t new_order 1 $DB_SIZE
```

---



---

## *table\_size.sh*

---



---

```
#!/usr/bin/sh
#*****
#*****
#@(#) Version: A.10.10 $Date: 97/12/15 10:49:57 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*****
#*****

isql -Usa -P <<- EOF

use tpcc
go

sp_spaceused history,1
go

sp_spaceused new_order,1
go

sp_spaceused orders,1
go

sp_spaceused order_line,1
go

sp_spaceused warehouse, 1
go

sp_spaceused district, 1
go

sp_spaceused customer, 1
go

sp_spaceused item, 1
go

sp_spaceused stock,1
go

EOF
```

---



---

## *devcreate.sh*

---



---

```
#!/usr/bin/sh
#*****
#*****
#@(#) Version: A.10.10 $Date: 97/12/15 13:16:43 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*****
#*****

#@(#) devcreate.sh 1.1 6/7/95
#
```

```
# scripts/devcreate.sh
#
# Read a device file from stdin in the format given in format/devices
# and output on stdout the SQL statements to create the devices,
# databases and segments defined by the input device file.
#
# The SQL is output in the following order
#
# 1) Disk inits and disk mirrors
# 2) Create databases
# 3) sp_addsegments and sp_extendsegments
#
if [ "$1" = "buildmaster" ]
then
    bm=y
    bmbinary=$2
elif [ "$1" = "sql" ]
then
    bm=n
    release=$2
else
    echo "Usage : $0 [buildmaster buildmaster_binary |sql] < device_file" >&2
    echo "buildmaster - generate buildmaster command" >&2
    echo "sql [release] - generate SQL commands" >&2
    exit 1
fi

in_device=n
in_db=n

logical_name=
physical_name=
device_size=

vdevno=0

sql_file=/tmp/dvsql$$
db_file=/tmp/dvdb$$
db_s_file=/tmp/dvdb_s$$
seg_file=/tmp/dvseg$$
export sql_file db_file seg_file

grep -v '^#' | tr -s '\011 ' '\012\012' | while read token garbage
do
    case $token in
        DEVICE) # A new device
            # clear the fields for the next device
            logical_name=
            physical_name=
            device_size=
            vstart_offset=
            mirror=

            in_device=y
            in_db=n
            ;;

        db=*) # database name
            if [ "$in_db" = "y" ]
            then
                # Store info about db
                echo

            fi

            # Start the new database
            db_name="echo $token | sed 's/db=/'"
            db_log=
            db_size=0

            in_db=y
            ;;

        log) # This disk is the log disk for the current db
            if [ "$in_db" = "y" ]
            then
                db_log="log on"

            fi
    esac
done
```

```

;;
vstart=*)
vstart_offset=`echo $token | sed 's/vstart=/'`
;;
mirror=*)
# store info about log-mirror
mirror=`echo $token | sed 's/mirror=/'`
;;
size=*) # the size of the current db on this disk
if [ "$in_db" = "y" ]
then
# Store info about size
db_size=`echo $token | sed 's/size=/'`
fi
;;
segment=*)
if [ "$in_db" = "y" ]
then
segment=`echo $token | sed 's/segment=/'`
echo "$db_name $segment $logical_name" >>
$seg_file
fi
;;
DEVICE_END)# Complete the device
# Save any database fragment information
if [ "$in_db" = "y" ]
then
echo "$db_name $logical_name $db_size
$db_log" >> $db_file
fi
if [ "$b" = "y" -a "$in_device" = "y" -a "$logical_name"
= "master" ]
then
# Convert Mb to 2k pages.
page_size=`expr $device_size \/* 512`
echo "$b binary -d$physical_name -
s$page_size"
fi
# The disk init SQL, but not for the master device
#
if [ "$b" = "n" -a "$in_device" = "y" -a "$logical_name"
!= "master" ]
then
# Convert Mb to 2k pages.
page_size=`expr $device_size \/* 512`
# echo SQL to create the device
echo "disk init"
echo " name = '$logical_name',"
echo " physname = '$physical_name',"
echo " vdevno = $vdevno,"
echo " size = $page_size"
if [ "$vstart_offset" != "" ]
then
echo " , vstart = $vstart_offset"
fi
echo go
fi
# The disk mirror SQL (including master)
if [ "$mirror" = "" -o "$b" = "y" ]
then
false;
else
# Echo SQL to create the disk mirror
echo "disk mirror"
echo " name = '$logical_name',"
echo " mirror = '$mirror',"
echo " writes = noserial"
echo go
fi
fi
*) # could be one of several
if [ "$in_device" = "y" ]
then
if [ "$logical_name" = "" ]
then
logical_name=$token
if [ $logical_name != "master" ]
then
vdevno=`expr $vdevno +
1`
fi
elif [ "$physical_name" = "" ]
then
physical_name=$token
elif [ "$device_size" = "" ]
then
device_size=$token
else
echo
fi
else
echo
fi
fi
esac
done
# If we are in buildmaster mode we can just stop here.
if [ "$b" = "y" ]
then
rm $db_file $seg_file
exit 0
fi
#
# Now we have generated the disk init commands, create the
# create database commands.
#
# The file $db_file will have been created with the following format
#
# dbname device size [log on]
#
#sort $db_file > $db_s_file
cat $db_file > $db_s_file
rm $db_file
# Add a dummy line end to the database file
echo "__$$" >> $db_s_file
current_db=
in_db=n
logdbinfo=
export in_db current_db logdbinfo
cat $db_s_file | while read dbname device size log
do
if [ "$dbname" = "$current_db" ]
then
if [ -z "$log" ]
then
if [ -z "$dbinfo" ]
then
dbinfo="on $device = $size"
else
dbinfo="$dbinfo, $device = $size"
fi
else
if [ -z "$logdbinfo" ]
then
logdbinfo="$log $device = $size"
else
logdbinfo="$logdbinfo, $device = $size"
fi
fi
fi
elif [ "$in_db" = "y" ]
then

```

```

echo "create database $current_db"
echo $dbinfo
if [ -n "$logdbinfo" ]
then
    echo $logdbinfo
fi
echo go
logdbinfo=

current_db=$dbname
dbinfo="on $device = $size"
in_db=y
else
current_db=$dbname
if [ -z "$log" ]
then
    dbinfo="on $device = $size"
else
    logdbinfo="$log $device = $size"
fi
in_db=y
fi
done
#rm $db_s_file

#
# Now we have the create database commands, create the segment commands
#
# The file $seg_file will have been created with the following format
#
# dbname device segment
#

current_db=
current_seg=
seg_db=
export current_seg current_db seg_db

sort $seg_file | while read dbname segment device garbage
do
    if [ "$dbname" = "$current_db" ]
    then
        false
    else
        echo "use $dbname"
        echo go
        # In System 10 segment procs now takes db as 2nd arg
        if [ "$release" = "System10" ]
        then
            seg_db="$dbname ,"
        fi
    fi

    if [ "$segment" = "system" -o "$segment" = "default" ]
    then
        false # do nothing
    elif [ "$segment" = "$current_seg" ]
    then
        echo "sp_extendsegment $segment , $seg_db $device"
        echo go
    else
        echo "sp_addsegment $segment , $seg_db $device"
        echo go
    fi
    current_seg=$segment
    current_db=$dbname
done

# now sort the segment file in database, device order
# to enable us to drop the unwanted system and default segments

in_device=no
export in_device
sort +0 -1 +2 -3 $seg_file | while read dbname segment device garbage
do
    if [ "$device" = "$current_dev" ]
    then
        false
    else

```

```

if [ "$in_device" = "yes" ]
then
    if [ "$drop_segs" = "yes" ]
    then
        echo "sp_dropsegment 'default', $seg_db"
        echo go
        echo "sp_dropsegment 'system', $seg_db"
        echo go
    fi
fi

in_device=yes
drop_segs=yes

if [ "$dbname" = "$current_db" ]
then
    false
else
    echo "use $dbname"
    echo go
    # In System 10 segment procs now takes db as 2nd arg
    if [ "$release" = "System10" ]
    then
        seg_db="$dbname ,"
    fi
fi

if [ "$segment" = "system" -o "$segment" = "default" ]
then
    drop_segs=no
fi

current_dev=$device
current_db=$dbname
done

rm $seg_file

echo "use master"
echo go
echo "checkpoint"
echo go

```

---



---

## *tpcc\_tables.sh*

---



---

```

#!/usr/bin/sh -f
*****
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****

isql -Usa -P$PASSWORD << EOF
/* This script will create all the tables required for TPC-C benchmark */
/* It will also create some of the indexes. */
sp_dboption tpcc,"select into/bulkcopy",true
go
use tpcc
go
checkpoint
go

if exists ( select name from sysobjects where name = 'history' )
    alter table history unpartition
go
if exists ( select name from sysobjects where name = 'history' )

```



```

drop table history
go
if exists ( select name from sysobjects where name = 'orders' )
drop table orders
go
if exists ( select name from sysobjects where name = 'new_order' )
drop table new_order
go
if exists ( select name from sysobjects where name = 'item' )
drop table item
go
if exists ( select name from sysobjects where name = 'district' )
drop table district
go
if exists ( select name from sysobjects where name = 'warehouse' )
drop table warehouse
go
if exists ( select name from sysobjects where name = 'order_line' )
drop table order_line
go
create table order_line (
    ol_o_id      int,
    ol_d_id      tinyint,
    ol_w_id      smallint,
    ol_number    tinyint,
    ol_i_id      int,
    ol_supply_w_id smallint,
    ol_delivery_d datetime, /*- Updated by D */
    ol_quantity  smallint,
    ol_amount    float,
    ol_dist_info char(24)
) on Sorder_line
go
create unique clustered index ol_clu
on order_line(ol_w_id, ol_d_id, ol_o_id, ol_number)
on Sorder_line
go
dbcc tune(ascinserts, 1, order_line)
go
dbcc tune(oamtrips, 100, order_line)
go
create table warehouse (
    w_id      smallint,
    w_name    char(10),
    w_street_1 char(20),
    w_street_2 char(20),
    w_city    char(20),
    w_state   char(2),
    w_zip     char(9),
    w_tax     real,
    w_ytd    float /*- Updated by PID, PNM */
) with max_rows_per_page = 1 on Swarehouse
go
create table district (
    d_id      tinyint,
    d_w_id    smallint,
    d_name    char(10),
    d_street_1 char(20),
    d_street_2 char(20),
    d_city    char(20),
    d_state   char(2),
    d_zip     char(9),
    d_tax     real,
    d_ytd    float, /*- Updated by PID, PNM */
    d_next_o_id int /*- Updated by NO */
) with max_rows_per_page = 10 on Sdistrict
go
create table item (
    i_id      int,
    i_im_id   int,
    i_name    char(24),
    i_price   float,
    i_data    char(50)
) on Sitem
go
if exists ( select name from sysobjects where name = 'customer' )
drop table customer
go
create table customer (
    c_id      int,
    c_d_id    tinyint,
    c_w_id    smallint,
    c_first   char(16),
    c_middle  char(2),
    c_last    char(16),
    c_street_1 char(20),
    c_street_2 char(20),
    c_city    char(20),
    c_state   char(2),
    c_zip     char(9),
    c_phone   char(16),
    c_since   datetime,
    c_credit  char(2),
    c_credit_lim numeric(12),
    c_discount real,
    c_delivery_cnt smallint,
    c_payment_cnt smallint, /*- Updated by PNM, PID */
    c_balance float, /*- Updated by PNM, PID */
    c_ytd_payment float, /*- Updated by PNM, PID */
    c_data1    char(250), /*- Updated (?) by PNM, PID */
    c_data2    char(250) /*- Updated (?) by PNM, PID */
) on Scustomer
go
create unique clustered index c_clu
on customer(c_w_id, c_d_id, c_id)
on Scustomer
go
create table history (
    h_c_id      int,
    h_c_d_id    tinyint,
    h_c_w_id    smallint,
    h_d_id      tinyint,
    h_w_id      smallint,
    h_date      datetime,
    h_amount    float,
    h_data      char(24)
) on Shistory
go
alter table history partition 512
go
create table new_order (
    no_o_id      int,
    no_d_id      tinyint,
    no_w_id      smallint,
) on Snew_order
go
create unique clustered index no_clu
on new_order(no_w_id, no_d_id, no_o_id)
on Snew_order
go
dbcc tune(ascinserts, 1, new_order)
go
dbcc tune(oamtrips, 100, new_order)
go
create table orders (
    o_id      int,
    o_c_id    int,
    o_d_id    tinyint,
    o_w_id    smallint,
    o_entry_d datetime,
    o_carrier_id smallint, /*- Updated by D */

```

```

o_ol_cnt tinyint,
o_all_localtinyint
) on Sorders
go

create unique clustered index o_clu
on orders(o_w_id, o_d_id, o_id)
on Sorders

go
dbcc tune(ascinserts, 1, orders)
go
dbcc tune(oamtrips, 100, orders)
go

if exists ( select name from sysobjects where name = 'stock' )
drop table stock

go
create table stock (
s_i_id int,
s_w_id smallint,
s_quantitysmallint, /*- Updated by NO */
s_ytd int, /*- Updated by NO */
s_order_cntsmallint,/*- Updated by NO */
s_remote_cntsmallint,/*- Updated by NO */
s_dist_01 char(24),
s_dist_02 char(24),
s_dist_03 char(24),
s_dist_04 char(24),
s_dist_05 char(24),
s_dist_06 char(24),
s_dist_07 char(24),
s_dist_08 char(24),
s_dist_09 char(24),
s_dist_10 char(24),
s_data char(50)
) on Sstock
go

create unique clustered index s_clu
on stock(s_i_id, s_w_id)
on Sstock

go
dbcc tune(indextrips, 10, stock)
go

checkpoint
go
EOF

```

---



---

## *dumptran\_server.sh*

---



---

```

#!/usr/bin/sh -f
#####
#####
#@(#) Version: A.10.10 $Date: 97/12/15 13:17:58 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#####
#####

isql -Usa -P$PASSWORD << EOF
dbcc tune(maxwritedes, 50)
go
dump tran $1 with truncate_only
go
use $1
go
checkpoint
go
dbcc tune(maxwritedes, 5)
go
EOF

```

---



---

## *tpcc\_load.sh*

---



---

```

#!/usr/bin/sh
#####
#####
#@(#) Version: A.10.10 $Date: 97/12/15 13:16:29 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#####
#####

inc=99999 # Load all at once, no partial loads anymore
set -x

load_stuff()
{
cur=$1
end=$2
shift
shift
if [ "$*" = "" ]
then
tables="customer stock orders new_order history district warehouse"
else
tables=$*
fi
}

setup_server()

# load tables for each warehouse
echo "Before while, incr = $inc, cur = $cur, end = $end"
while [ `expr $cur + $inc` -lt $end ]
do
echo "In while, incr = $inc, cur = $cur, end = $end"
load_tables $cur `expr $cur + $inc - 1` $tables
cur=`expr $cur + $inc`
done

echo "Last step incr = $inc, cur = $cur, end = $end"
load_tables $cur $end $tables

}

load_tables()
{
first=$1; last=$2; shift;shift; tables1=$*

# load items with first warehouse
if [ $first -eq 1 ]
then
load -t item &
sleep 5
fi

# load each specified table
echo "loading $tables for warehouse $first thru $last"
for table in $tables1
do
echo "loading table $table for warehouse $first thru $last"
load -t $table $first $last &
sleep 5
done
wait
}

setup_server()
{
isql -Usa -P << EOF
use tpcc

```

```

go
dbcc iosize("tpcc", "new_order", 16)
go
dbcc iosize("tpcc", "order_line", 16)
go
dbcc iosize("tpcc", "orders", 16)
go
dbcc iosize("tpcc", "stock", 16)
go
dbcc iosize("tpcc", "item", 16)
go
dbcc iosize("tpcc", "customer", 16)
go
dbcc iosize("tpcc", "history", 16)
go
EOF
}

```

```
load_stuff $*
```

---



---

## run\_server

---



---

```

#!/usr/bin/sh
#####
#####
#@ (#) Version: A.10.10 $Date: 97/12/15 10:49:56 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#####
#####
#
# The default dataserver is the one from the release bin.
#
dataserver=$SQL_RELEASE/bin/dataserver
server_options="-c${CONFIG_FILE}"

# Do we override this ?
if [ $# != 0 ]
then
  if [ "$1" != "-" ]
  then
    dataserver=$1
  fi

  # Pick up the remaining arguments.
  shift
  server_options="$server_options" $*"
fi

(cd ~tpcc/dev; $dataserver -d$MASTER_DEVICE $server_options)&

touch ~tpcc/dev/errorlog; tail -f -c1 ~tpcc/dev/errorlog | grep -q 'Recovery complete'

```

---



---

## shutdown\_server.sh

---



---

```

#!/usr/bin/sh -f
#####
#####
#@ (#) Version: A.10.10 $Date: 97/12/15 10:49:57 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#####
#####

```

```

# Now shut the server down
isql -Usa -PSPASSWORD <<EOF
shutdown
go
EOF

```

```

# The server may not be completely shutdown yet.
# Wait a bit longer just to be sure.
# (We really would like some guarantee)
sleep 20

```

---



---

## devices

---



---

```

DEVICE master master 250 db=tpcc size=250 segment=default segment=system
DEVICE_END
DEVICE master2 master2 50 db=tpcc size=50 segment=default segment=system
DEVICE_END

```

```
DEVICE log1 log1 20000 db=tpcc size=20000 log DEVICE_END
```

```

DEVICE w_d_i_no w_d_i_no 600 db=tpcc size=600 segment=Swarehouse segment=Sdistrict segment=Sitem segment=Snew_order DEVICE_END

```

```

DEVICE customer1 customer1 7380 db=tpcc size=7380 segment=Scustomer
DEVICE_END
DEVICE customer2 customer2 7380 db=tpcc size=7380 segment=Scustomer
DEVICE_END
DEVICE customer3 customer3 7380 db=tpcc size=7380 segment=Scustomer
DEVICE_END
DEVICE customer4 customer4 7380 db=tpcc size=7380 segment=Scustomer
DEVICE_END
DEVICE customer5 customer5 7380 db=tpcc size=7380 segment=Scustomer
DEVICE_END
DEVICE customer6 customer6 7380 db=tpcc size=7380 segment=Scustomer
DEVICE_END
DEVICE customer7 customer7 7380 db=tpcc size=7380 segment=Scustomer
DEVICE_END
DEVICE customer8 customer8 7380 db=tpcc size=7380 segment=Scustomer
DEVICE_END
DEVICE customer9 customer9 7380 db=tpcc size=7380 segment=Scustomer
DEVICE_END
DEVICE customer10 customer10 7380 db=tpcc size=7380 segment=Scustomer
DEVICE_END
DEVICE customer11 customer11 7380 db=tpcc size=7380 segment=Scustomer
DEVICE_END
DEVICE customer12 customer12 7380 db=tpcc size=7380 segment=Scustomer
DEVICE_END
DEVICE customer13 customer13 4500 db=tpcc size=4500 segment=Scustomer
DEVICE_END

```

```

DEVICE orders1 orders1 345 db=tpcc size=345 segment=Sorders DEVICE_END
DEVICE orders2 orders2 345 db=tpcc size=345 segment=Sorders DEVICE_END
DEVICE orders3 orders3 345 db=tpcc size=345 segment=Sorders DEVICE_END
DEVICE orders4 orders4 345 db=tpcc size=345 segment=Sorders DEVICE_END
DEVICE orders5 orders5 345 db=tpcc size=345 segment=Sorders DEVICE_END
DEVICE orders6 orders6 345 db=tpcc size=345 segment=Sorders DEVICE_END
DEVICE orders7 orders7 345 db=tpcc size=345 segment=Sorders DEVICE_END
DEVICE orders8 orders8 345 db=tpcc size=345 segment=Sorders DEVICE_END
DEVICE orders9 orders9 345 db=tpcc size=345 segment=Sorders DEVICE_END
DEVICE orders10 orders10 345 db=tpcc size=345 segment=Sorders DEVICE_END
DEVICE orders11 orders11 800 db=tpcc size=800 segment=Sorders DEVICE_END
DEVICE orders12 orders12 200 db=tpcc size=200 segment=Sorders DEVICE_END

```

```

DEVICE order_line1 order_line1 7735 db=tpcc size=7735 segment=Sorder_line
DEVICE_END
DEVICE order_line2 order_line2 7735 db=tpcc size=7735 segment=Sorder_line
DEVICE_END
DEVICE order_line3 order_line3 7735 db=tpcc size=7735 segment=Sorder_line
DEVICE_END
DEVICE order_line4 order_line4 7735 db=tpcc size=7735 segment=Sorder_line
DEVICE_END
DEVICE order_line5 order_line5 7735 db=tpcc size=7735 segment=Sorder_line
DEVICE_END
DEVICE order_line6 order_line6 7735 db=tpcc size=7735 segment=Sorder_line
DEVICE_END
DEVICE order_line7 order_line7 7735 db=tpcc size=7735 segment=Sorder_line
DEVICE_END

```

```

DEVICE order_line8 order_line8 7735 db=tpcc size=7735 segment=Sorder_line
DEVICE_END
DEVICE order_line9 order_line9 7735 db=tpcc size=7735 segment=Sorder_line
DEVICE_END
DEVICE order_line10 order_line10 7735 db=tpcc size=7735 segment=Sorder_line
DEVICE_END
DEVICE order_line11 order_line11 16500 db=tpcc size=16500 segment=Sorder_line
DEVICE_END
DEVICE order_line12 order_line12 800 db=tpcc size=800 segment=Sorder_line
DEVICE_END

DEVICE history1 history1 8400 db=tpcc size=8400 segment=Shistory DEVICE_END
DEVICE history2 history2 200 db=tpcc size=200 segment=Shistory DEVICE_END

DEVICE c_index1 c_index1 1024 db=tpcc size=1024 segment=Sc_index
DEVICE_END
DEVICE c_index2 c_index2 1024 db=tpcc size=1024 segment=Sc_index
DEVICE_END
DEVICE c_index3 c_index3 1024 db=tpcc size=1024 segment=Sc_index
DEVICE_END
DEVICE c_index4 c_index4 1024 db=tpcc size=1024 segment=Sc_index
DEVICE_END
DEVICE c_index5 c_index5 1024 db=tpcc size=1024 segment=Sc_index
DEVICE_END
DEVICE c_index6 c_index6 1024 db=tpcc size=1024 segment=Sc_index
DEVICE_END
DEVICE c_index7 c_index7 1500 db=tpcc size=1500 segment=Sc_index
DEVICE_END
DEVICE c_index8 c_index8 4000 db=tpcc size=4000 segment=Sc_index
DEVICE_END

DEVICE stock1 stock1 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock2 stock2 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock3 stock3 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock4 stock4 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock5 stock5 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock6 stock6 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock7 stock7 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock8 stock8 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock9 stock9 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock10 stock10 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock11 stock11 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock12 stock12 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock13 stock13 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock14 stock14 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock15 stock15 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock16 stock16 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock17 stock17 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock18 stock18 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock19 stock19 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock20 stock20 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock21 stock21 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock22 stock22 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock23 stock23 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock24 stock24 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock25 stock25 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock26 stock26 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock27 stock27 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock28 stock28 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock29 stock29 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock30 stock30 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock31 stock31 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock32 stock32 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock33 stock33 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock34 stock34 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock35 stock35 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock36 stock36 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock37 stock37 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock38 stock38 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock39 stock39 3930 db=tpcc size=3930 segment=Sstock DEVICE_END
DEVICE stock40 stock40 7665 db=tpcc size=7665 segment=Sstock DEVICE_END

```

## B.2 Code to Populate

### load.c

```

/*****
*****
@(#) Version: A.10.10 $Date: 97/12/15 14:06:16 $

(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****
*****/

/*****
****
*****
****

To Do:
o Need to add CLAST and CID constants and way to set them

*****
****
*****
*****/

#include <unistd.h>
#include <time.h>
#include <stdio.h>
#include "tpcc.h"
#include "random.h"

/* configurable parameters */
#define NURAND_C 123

typedef unsigned long BitVector;
#define WSZ (sizeof(BitVector)*8)
#define nthbit(map,n) map[(n)/WSZ] & (((BitVector)0x1)<< ((n)%WSZ))
#define setbit(map,n) map[(n)/WSZ] |= (((BitVector)0x1)<< ((n)%WSZ))

int load_item;
int load_warehouse;
int load_district;
int load_history;
int load_orders;
int load_orderline;
int load_neworder;
int load_customer;
int load_stock;

ID first;
ID last;

void LoadWarehouse();
void LoadDistrict();
void LoadItems();

int main(argn, argv)
int argn;
char **argv;
{
ID w_id;

configure(argn, argv);
begin_load();

/* NOTE: Orders and Orderline must have the same seed to work
properly */
if (load_item)
{ SetRandomSeed(100); InitRandomStrings(); LoadItems(); }
if (load_warehouse)
{ SetRandomSeed(101); InitRandomStrings(); LoadWarehouse(first, last); }
if (load_district)
{ SetRandomSeed(102); InitRandomStrings(); LoadDistrict(first, last); }
if (load_stock)
{ SetRandomSeed(103); InitRandomStrings(); LoadStock(first, last); }
if (load_customer)
{ SetRandomSeed(104); InitRandomStrings(); LoadCustomer(first, last); }

```

```

if (load_history)
  { SetRandomSeed(105); InitRandomStrings(); LoadHist(first, last); }
if (load_orders)
  { SetRandomSeed(106); InitRandomStrings(); LoadOrders(first, last); }
if (load_orderline)
  { SetRandomSeed(106); InitRandomStrings(); LoadOrderLine(first, last); }
if (load_neworder)
  { SetRandomSeed(107); InitRandomStrings(); LoadNeworder(first, last); }

end_load();

return 0;
}

/*****
*
*****/

Warehouse

/*****
***/

void
LoadWarehouse(first, last)
  ID first, last;
  {
  warehouse_row r[1];
  ID w_id;

  begin_warehouse_load();

  printf("loading warehouses %d to %d\n",first,last);
  r->W_YTD = 30000000;
  for (w_id = first; w_id <= last; w_id++)
    {
    printf("loading warehouse %d\n",w_id);

    r->W_ID = w_id;
    MakeAlphaString(6, 10, r->W_NAME);
    MakeAddress(r->W_STREET_1, r->W_STREET_2, r->W_CITY, r->W_STATE,
               r->W_ZIP);
    r->W_TAX = RandomNumber(0, 2000) / 10000.0;

    warehouse_load(r);
    }

  end_warehouse_load();
}

/*****
***/

District

/*****
***/

void
LoadDistrict(first, last)
  ID first, last;
  {
  ID w_id, d_id;
  district_row r[1];

  begin_district_load();

  r->D_YTD = 3000000;
  r->D_NEXT_O_ID = 3001;

  for (w_id = first; w_id <= last; w_id++)

```

```

  {
  printf("loading districts for warehouse %d\n",w_id);
  r->D_W_ID = w_id;

  for (d_id = 1; d_id <= DIST_PER_WARE; d_id++)
    {
    r->D_ID = d_id;
    MakeAlphaString(6, 10, r->D_NAME);
    MakeAddress(r->D_STREET_1, r->D_STREET_2, r->D_CITY, r->D_STATE,
               r->D_ZIP);
    r->D_TAX = RandomNumber(0, 2000) / 10000.0;

    district_load(r);
    }
  }

end_district_load();
}

/*****
***/

Item

/*****
***/

void
LoadItems()
  {
  item_row r[1];
  int perm[MAXITEMS+1];
  ID i_id;

  begin_item_load();

  /* select exactly 10% of items to be labeled "original" */
  RandomPermutation(perm, MAXITEMS);

  /* do for each item */
  printf("loading item table\n");
  for (i_id = 1; i_id <= MAXITEMS; i_id++)
    {
    /* Generate Item Data */
    r->I_ID = i_id;
    MakeAlphaString(14, 24, r->I_NAME);
    r->I_PRICE = RandomNumber(100,10000);
    MakeAlphaString(26, 50, r->I_DATA);
    if (perm[r->I_ID] <= (MAXITEMS+9)/10)
      Original(r->I_DATA);
    r->I_IM_ID = RandomNumber(1, 10000);

    item_load(r);
    }
  }

end_item_load();
}

/*****
***/

History

/*****
***/

LoadHist(first, last)
  ID first, last;
  {
  ID w_id, d_id, c_id;

```

```

static history_row r;

begin_history_load();

for (w_id = first; w_id <= last; w_id++) {
    printf("Loading history for warehouse %d\n",w_id);
    for (d_id = 1; d_id <= DIST_PER_WARE; d_id++) {
        for (c_id = 1; c_id <= CUST_PER_DIST; c_id++) {
            r.H_C_D_ID = r.H_D_ID = d_id;
            r.H_C_W_ID = r.H_W_ID = w_id;
            r.H_C_ID = c_id;
            CurrentDate(&r.H_DATE);
            r.H_AMOUNT = 1000;
            SelectHistoryData(r.H_DATA);
            history_load(&r);
        }
    }
}
end_history_load();
}

/*****
*****/

Customer

/*****
*****/

LoadCustomer(first, last)
    ID first, last;
{
    ID w_id;

    begin_customer_load();

    for (w_id = first; w_id <= last; w_id++) {
        printf("Loading customer for warehouse %d\n",w_id);
        Customer(w_id);
    }

    end_customer_load();
}

Customer(w_id)
/*****
*****/
Load customers for the given warehouse and district
/*****
*****/
    ID w_id;
    {
        int i;
        ID id[CUST_PER_DIST+1];
        ID c_id;
        ID d_id;
        customer_row r[1];
        static int bad_credit_perm[DIST_PER_WARE+1][CUST_PER_DIST+1];

        /* 10% of customers will have bad credit */
        for (d_id = 1; d_id <= DIST_PER_WARE; d_id++)
            RandomPermutation(bad_credit_perm[d_id], CUST_PER_DIST);

        /* Order by customer id, then district */
        r->C_W_ID = w_id;
        r->C_CREDIT_LIM = 5000000;
        r->C_BALANCE = -1000;
        r->C_YTD_PAYMENT = 1000;
        r->C_PAYMENT_CNT = 1;
        r->C_DELIVERY_CNT = 0;
        for (c_id=1; c_id <= CUST_PER_DIST; c_id++)
            {
                r->C_ID = c_id;
                for (d_id = 1; d_id <= DIST_PER_WARE; d_id++)
                    {
                        r->C_D_ID = d_id;

                        if (c_id <= 1000)
                            LastName(c_id - 1, r->C_LAST);
                        else
                            LastName(NURandomNumber(255, 0, 999, NURAND_C), r->C_LAST);

                        strcpy(r->C_MIDDLE, "OE");
                        SelectFirstName(r->C_FIRST);
                        MakeAddress(r->C_STREET_1, r->C_STREET_2, r->C_CITY, r->C_STATE,
                            r->C_ZIP);
                        SelectPhoneData(r->C_PHONE);
                        CurrentDate(&r->C_SINCE);
                        if (bad_credit_perm[d_id][r->C_ID] <= ((CUST_PER_DIST + 9)/10))
                            strcpy(r->C_CREDIT, "BC");
                        else
                            strcpy(r->C_CREDIT, "GC");
                        r->C_DISCOUNT = RandomNumber(0, 5000) / 10000.0;
                        SelectClientData(r->C_DATA);

                        customer_load(r);
                    }
                }
            }

/*****
*****/

Order, Order line, New order

/*****
*****/

LoadOrders(first, last)
    ID first, last;
    {
        ID w_id, d_id;

        begin_order_load();

        for (w_id = first; w_id <= last; w_id++) {
            printf("Loading Orders for warehouse %d\n",w_id);
            for (d_id = 1; d_id <= DIST_PER_WARE; d_id++) {
                Orders(w_id, d_id);
            }
        }
        end_order_load();
    }

LoadOrderLine(first, last)
    ID first, last;
    {
        ID w_id, d_id;

        begin_orderline_load();

        for (w_id = first; w_id <= last; w_id++) {
            printf("Loading Orderline for warehouse %d\n",w_id);
            for (d_id = 1; d_id <= DIST_PER_WARE; d_id++) {
                OrderLine(w_id, d_id);
            }
        }
        end_orderline_load();
    }

Orders(w_id, d_id)
    ID w_id, d_id;
    {
        int cust[ORD_PER_DIST+1];
        ID o_id, ol_number;
        ID ol;
        order_row r[1];
        orderline_row olr[1];
        int sum;

        r->O_W_ID = w_id;

```

```

r->O_D_ID = d_id;

RandomPermutation(cust, ORD_PER_DIST);

r->O_ALL_LOCAL = 1;
olr->OL_QUANTITY = 5;
for (o_id = 1; o_id <= ORD_PER_DIST; o_id++) {
    r->O_ID = o_id;
    r->O_C_ID = cust[o_id];
    CurrentDate(&r->O_ENTRY_D);

    if (r->O_ID <= 2100) r->O_CARRIER_ID = RandomNumber(1,10);
    else r->O_CARRIER_ID = EMPTY_NUM;

    /* map the range 1..n onto 5..15 for orderline count */
    r->O_OL_CNT = RandomNumber(5,15);

    /* load the order */
    order_load(r);

    /* generate the order lines */
    olr->OL_O_ID = o_id;
    olr->OL_D_ID = d_id;
    olr->OL_W_ID = w_id;
    olr->OL_SUPPLY_W_ID = w_id;
    for (ol_number = 1; ol_number <= r->O_OL_CNT; ol_number++) {
        olr->OL_NUMBER = ol_number;
        olr->OL_I_ID = RandomNumber(1, MAXITEMS);

        /* Store null CurrentDate in the DB as "01/01/1800 12:00:00AM" */
        if (o_id <= 2100) {
            olr->OL_DELIVERY_D = *(&r->O_ENTRY_D);
            olr->OL_AMOUNT = 0;
        } else {
            EmptyDate(&olr->OL_DELIVERY_D);
            olr->OL_AMOUNT = RandomNumber(1, 999999);
        }

        SelectStockDistrict(olr->OL_DIST_INFO);
    }
}

OrderLine(w_id, d_id)
ID w_id, d_id;
{
    int cust[ORD_PER_DIST+1];
    ID o_id, ol_number;
    ID ol;
    order_row r[1];
    orderline_row olr[1];
    int sum;

    r->O_W_ID = w_id;
    r->O_D_ID = d_id;

    RandomPermutation(cust, ORD_PER_DIST);

    r->O_ALL_LOCAL = 1;
    olr->OL_QUANTITY = 5;
    for (o_id = 1; o_id <= ORD_PER_DIST; o_id++) {
        r->O_ID = o_id;
        r->O_C_ID = cust[o_id];
        CurrentDate(&r->O_ENTRY_D);

        if (r->O_ID <= 2100) r->O_CARRIER_ID = RandomNumber(1,10);
        else r->O_CARRIER_ID = EMPTY_NUM;

        /* map the range 1..n onto 5..15 for orderline count */
        r->O_OL_CNT = RandomNumber(5,15);

        /* generate the order lines */
        olr->OL_O_ID = o_id;
        olr->OL_D_ID = d_id;
        olr->OL_W_ID = w_id;
        olr->OL_SUPPLY_W_ID = w_id;
        for (ol_number = 1; ol_number <= r->O_OL_CNT; ol_number++) {
            olr->OL_NUMBER = ol_number;

            olr->OL_I_ID = RandomNumber(1, MAXITEMS);

            /* Store null CurrentDate in the DB as "01/01/1800 12:00:00AM" */
            if (o_id <= 2100) {
                olr->OL_DELIVERY_D = *(&r->O_ENTRY_D);
                olr->OL_AMOUNT = 0;
            } else {
                EmptyDate(&olr->OL_DELIVERY_D);
                olr->OL_AMOUNT = RandomNumber(1, 999999);
            }

            SelectStockDistrict(olr->OL_DIST_INFO);

            orderline_load(olr);
        }
    }

    LoadNeworder(first, last)
    ID first, last;
    {
        ID w_id, d_id;

        begin_neworder_load();

        for (w_id = first; w_id <= last; w_id++) {
            printf("Loading NewOrder for warehouse %d\n", w_id);
            for (d_id = 1; d_id <= DIST_PER_WARE; d_id++) {
                neworder_row r[1];

                r->NO_D_ID = d_id;
                r->NO_W_ID = w_id;
                for (r->NO_O_ID=2101; r->NO_O_ID <= ORD_PER_DIST; r->NO_O_ID++)
                {
                    neworder_load(r);
                }
            }
        }

        end_neworder_load();
    }

#define ITEM_BITVEC_SIZE ((MAXITEMS/(8*sizeof(BitVector)))+1)*sizeof(BitVector)

LoadStock(first, last)
ID first, last;
{
    BitVector **perm;
    stock_row r[1];
    ID w_id;
    ID i_id;
    unsigned long count = 0;
    unsigned long checkPointTime, checkPointChunk, totalRowsToLoad;
    int i;
    long j;

    begin_stock_load();

    perm = (BitVector **) malloc((last-first)*sizeof(BitVector *));
    if (perm == NULL) {
        perror("LoadStock: can't allocate memory for permutations\n");
    }
    /* select exactly 10% of items to be labeled "original" */
    for (w_id = first; w_id <= last; w_id++)
    {
        int index = w_id - first;
        perm[index] = (BitVector *)malloc(ITEM_BITVEC_SIZE);
        if (perm[index] == NULL) {
            perror("LoadStock: can't allocate memory\n");
        }
        (void) memset(perm[index], 0, ITEM_BITVEC_SIZE);
        /* Mark exactly 10% of items as "original" */
        for (i = 0; i < (MAXITEMS+9)/10; i++) {
            do {
                j = RandomNumber(0, MAXITEMS-1);
            } while (nthbit(perm[index], j));
            setbit(perm[index], j);
        }
    }
}

```

```

    }
}

/* do for each item and warehouse */
printf("Loading stock items for warehouses %d to %d...\n",first, last);
r->S_YTD = 0;
r->S_ORDER_CNT = 0;
r->S_REMOTE_CNT = 0;
totalRowsToLoad = MAXITEMS*(last - first + 1);
/* Every 5% loaded, print message 8 */
checkPointTime = checkPointChunk = (totalRowsToLoad)/20;
for (i_id = 1; i_id <= MAXITEMS; i_id++) {
    r->S_I_ID = i_id;
    if (checkPointTime < count) {
        printf("Loaded %4.1lf%% of stock rows\n",100.0*((double)count/(double)totalRowsToLoad));
        checkPointTime += checkPointChunk;
    }
    for (w_id = first; w_id <= last; w_id++) {
        /* Generate Stock Data */
        r->S_W_ID = w_id;
        r->S_QUANTITY = RandomNumber(10,100);
        SelectStockDistrict(r->S_DIST_01);
        SelectStockDistrict(r->S_DIST_02);
        SelectStockDistrict(r->S_DIST_03);
        SelectStockDistrict(r->S_DIST_04);
        SelectStockDistrict(r->S_DIST_05);
        SelectStockDistrict(r->S_DIST_06);
        SelectStockDistrict(r->S_DIST_07);
        SelectStockDistrict(r->S_DIST_08);
        SelectStockDistrict(r->S_DIST_09);
        SelectStockDistrict(r->S_DIST_10);
        SelectStockData(r->S_DATA);
        if (nthbit(perm[w_id - first], r->S_I_ID - 1) {
            Original(r->S_DATA);
        }
        stock_load(r);
        count++;
    }
}
printf("finished loading stock items for warehouses %d to %d\n",
    first,last);

for (w_id = first; w_id <= last; w_id++) {
    free(perm[w_id-first]);
}

free(perm);

end_stock_load();
}

configure(argc, argv)
/*****
configure configures the load stuff
By default, loads all the tables for a the specified warehouse.
When loading warehouse 1, also loads the item table.
*****/
int argc;
char **argv;
{
    char ch;
    int any_except_item, any_at_all;

    /* use unbuffered I/O (for output to files) */
    setvbuf(stdout, 0, _IONBF, 0);
    setvbuf(stderr, 0, _IONBF, 0);

    /* define the defaults */
    load_item = load_warehouse = load_district = load_history =
        load_orders = load_orderline = load_neworder = load_customer = load_stock =
    NO;

    /* do for each option */
    while ((ch = getopt (argc, argv, "t:")) != EOF)

```

```

/* process according to options */
switch ( ch )
{
    /* check for TPC-A or TPC-B */
    case 't':
        if (strcmp(optarg, "warehouse") == 0)load_warehouse = YES;
        else if (strcmp(optarg, "district") == 0) load_district = YES;
        else if (strcmp(optarg, "stock") == 0) load_stock = YES;
        else if (strcmp(optarg, "item") == 0) load_item = YES;
        else if (strcmp(optarg, "history") == 0) load_history = YES;
        else if (strcmp(optarg, "orders") == 0) load_orders = YES;
        else if (strcmp(optarg, "orderline") == 0) load_orderline = YES;
        else if (strcmp(optarg, "new_order") == 0) load_neworder = YES;
        else if (strcmp(optarg, "customer") == 0) load_customer = YES;
        else
            error("%s is not a valid table name\n", optarg);
        continue;

        default: error("Bad runstring argument.\n");
            break;
    }

    /* some common flags depending on tables asked for */
    any_except_item = load_warehouse || load_district || load_stock ||
        load_history || load_orders || load_orderline ||
        load_neworder || load_customer;
    any_at_all = any_except_item || load_item;

    /* if only asked for item, don't allow warehouse to be specified */
    if (!any_except_item && load_item)
    {
        if (optind != argc)
            error("Don't specify warehouse when loading items");
    }

    /* otherwise get the warehouse number */
    else
    {
        if (optind >= argc)
            error("Must specify warehouses to load\n");
        first = atoi(argv[optind++]);

        if (optind >= argc)
            last = first;
        else
            last = atoi(argv[optind++]);

        if (first > last)
            error("First warehouse is greater than last warehouse\n");

        if (first <= 0)
            error("Warehouse must be positive non-zero\n");
    }

    /* if no tables mentioned explicitly, then load them all */
    if (!any_at_all)
    {
        load_warehouse = load_district = load_history = load_orders =
            load_orderline = load_neworder = load_customer = load_stock = YES;
        load_item = (first == 1);
    }
}

```

---



---

***bulk\_sybase.c***

---



---

```

/*****
*****

```



@(#) Version: A.10.10 \$Date: 97/12/15 10:57:22 \$

(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.  
\*\*\*\*\*  
\*\*\*\*\*/

/\*  
\*\*\*\*\*  
\*\*\*\*\*/

### Sybase Specific Routines

\*\*\*\*\*  
\*\*\*\*\*/

#define BATCH 1024

#include "tpcc.h"  
#include <sybfront.h>  
#include <sybdb.h>  
#include <syberror.h>

/\* various bind routines \*/  
#define bind\_ID bind\_int  
#define bind\_COUNT bind\_int  
#define bind\_LOGICAL bind\_int  
#define bind\_MONEY bind\_double  
#define bind\_REAL bind\_double  
#define bind\_TEXT bind\_str

#define load\_row(table) \  
{ \  
    (table)->count++; \  
    if (bcp\_sendrow((table)->dbproc) != SUCCEED) \  
        error("load\_row: Can't load row in %s table\n", (table)->name); \  
    if (table->count%BATCH == 0 && (bcp\_batch(table->dbproc) == -1)) \  
        error("load\_row: Can't post rows in %s table\n", (table)->name); \  
}

typedef struct  
{  
    int count;  
    char name[40];  
    DBPROCESS \*dbproc;  
} table\_info;

/\*  
\*\*\*

We rely on the static location row structures in the 'load' program so  
we only have to bind to the variables once.

\*\*\*\*\*  
\*\*\*\*\*/

table\_info w\_table[1];

begin\_warehouse\_load()  
{  
    table\_open(w\_table, "warehouse");  
}

warehouse\_load(r)  
    warehouse\_row \*r;  
{  
    static warehouse\_row \*old = NULL;

if (old != r)  
{  
    old = r;  
    bind\_ID(w\_table, 1, &r->W\_ID);  
    bind\_TEXT(w\_table, 2, r->W\_NAME);  
    bind\_TEXT(w\_table, 3, r->W\_STREET\_1);  
    bind\_TEXT(w\_table, 4, r->W\_STREET\_2);  
    bind\_TEXT(w\_table, 5, r->W\_CITY);  
    bind\_TEXT(w\_table, 6, r->W\_STATE);  
    bind\_TEXT(w\_table, 7, r->W\_ZIP);  
    bind\_REAL(w\_table, 8, &r->W\_TAX);  
    bind\_MONEY(w\_table, 9, &r->W\_YTD);  
}

load\_row(w\_table);  
}

end\_warehouse\_load()  
{  
    table\_close(w\_table);  
}

table\_info d\_table[1];

begin\_district\_load()  
{  
    table\_open(d\_table, "district");  
}

district\_load(r)  
    district\_row \*r;  
{  
    static district\_row \*old = NULL;

if (r != old)  
{  
    old = r;  
    bind\_ID(d\_table, 1, &r->D\_ID);  
    bind\_ID(d\_table, 2, &r->D\_W\_ID);  
    bind\_TEXT(d\_table, 3, r->D\_NAME);  
    bind\_TEXT(d\_table, 4, r->D\_STREET\_1);  
    bind\_TEXT(d\_table, 5, r->D\_STREET\_2);  
    bind\_TEXT(d\_table, 6, r->D\_CITY);  
    bind\_TEXT(d\_table, 7, r->D\_STATE);  
    bind\_TEXT(d\_table, 8, r->D\_ZIP);  
    bind\_REAL(d\_table, 9, &r->D\_TAX);  
    bind\_MONEY(d\_table, 10, &r->D\_YTD);  
    bind\_ID(d\_table, 11, &r->D\_NEXT\_O\_ID);  
}

load\_row(d\_table);  
}

end\_district\_load()  
{  
    table\_close(d\_table);  
}

table\_info i\_table[1];

begin\_item\_load()  
{  
    table\_open(i\_table, "item");  
}

item\_load(r)  
    item\_row \*r;  
{  
    static item\_row \*old = NULL;

if (r != old)  
{  
    old = r;  
    bind\_ID(i\_table, 1, &r->I\_ID);  
    bind\_ID(i\_table, 2, &r->I\_IM\_ID);  
    bind\_TEXT(i\_table, 3, r->I\_NAME);  
    bind\_MONEY(i\_table, 4, &r->I\_PRICE);  
    bind\_TEXT(i\_table, 5, r->I\_DATA);  
}

load\_row(i\_table);  
}

end\_item\_load()  
{  
    table\_close(i\_table);  
}

```

table_info h_table[1];

static DBDATETIME h_date;

begin_history_load()
{
    table_open(h_table, "history");
}

history_load(r)
    history_row *r;
{
    static history_row *old = NULL;

    if (r != old)
    {
        old = r;
        bind_ID(h_table, 1, &r->H_C_ID);
        bind_ID(h_table, 2, &r->H_C_D_ID);
        bind_ID(h_table, 3, &r->H_C_W_ID);
        bind_ID(h_table, 4, &r->H_D_ID);
        bind_ID(h_table, 5, &r->H_W_ID);
        bind_SYBDATE(h_table, 6, &h_date);
        bind_MONEY(h_table, 7, &r->H_AMOUNT);
        bind_TEXT(h_table, 8, r->H_DATA);
    }

    to_sybase_date(&r->H_DATE, &h_date);
    load_row(h_table);
}

end_history_load()
{
    table_close(h_table);
}

table_info c_table[1];

static char c_data_1[250+1];
static char c_data_2[250+1];

begin_customer_load()
{
    table_open(c_table, "customer");
}

customer_load(r)
    customer_row *r;
{
    int len;
    char *p;
    static customer_row *old = NULL;

    if (r != old)
    {
        old = r;
        bind_ID(c_table, 1, &r->C_ID);
        bind_ID(c_table, 2, &r->C_D_ID);
        bind_ID(c_table, 3, &r->C_W_ID);
        bind_TEXT(c_table, 4, r->C_FIRST);
        bind_TEXT(c_table, 5, r->C_MIDDLE);
        bind_TEXT(c_table, 6, r->C_LAST);
        bind_TEXT(c_table, 7, r->C_STREET_1);
        bind_TEXT(c_table, 8, r->C_STREET_2);
        bind_TEXT(c_table, 9, r->C_CITY);
        bind_TEXT(c_table, 10, r->C_STATE);
        bind_TEXT(c_table, 11, r->C_ZIP);
        bind_TEXT(c_table, 12, r->C_PHONE);
        bind_SYBDATE(c_table, 13, &r->C_SINCE);
        bind_TEXT(c_table, 14, r->C_CREDIT);

        bind_MONEY(c_table, 15, &r->C_CREDIT_LIM);
        bind_REAL(c_table, 16, &r->C_DISCOUNT);
        bind_COUNT(c_table, 17, &r->C_DELIVERY_CNT);
        bind_COUNT(c_table, 18, &r->C_PAYMENT_CNT);
        bind_MONEY(c_table, 19, &r->C_BALANCE);
        bind_MONEY(c_table, 20, &r->C_YTD_PAYMENT);
        bind_TEXT(c_table, 21, c_data_1);
        bind_TEXT(c_table, 22, c_data_2);
    }

    /* break up the data string into two pieces */
    len = strlen(r->C_DATA);
    if (len > 250)
    {
        memcpy(c_data_1, r->C_DATA, 250);
        c_data_1[250] = '\0';
        memcpy(c_data_2, r->C_DATA+250, len-250+1);
    }
    else
    {
        memcpy(c_data_1, r->C_DATA, 250+1);
        strcpy(c_data_2, "");
    }

    /* load the data */
    load_row(c_table);
}

end_customer_load()
{
    table_close(c_table);
}

table_info o_table[1];

DBDATETIME o_entry_d;
COUNT o_carrier_id;

begin_order_load()
{
    table_open(o_table, "orders");
}

order_load(r)
    order_row *r;
{
    static order_row *old = NULL;

    if (r != old)
    {
        old = r;
        bind_ID(o_table, 1, &r->O_ID);
        bind_ID(o_table, 2, &r->O_C_ID);
        bind_ID(o_table, 3, &r->O_D_ID);
        bind_ID(o_table, 4, &r->O_W_ID);
        bind_SYBDATE(o_table, 5, &o_entry_d);
        bind_ID(o_table, 6, &o_carrier_id);
        bind_COUNT(o_table, 7, &r->O_OL_CNT);
        bind_LOGICAL(o_table, 8, &r->O_ALL_LOCAL);
    }

    to_sybase_date(&r->O_ENTRY_D, &o_entry_d);
    if (r->O_CARRIER_ID == EMPTY_NUM) o_carrier_id = -1;
    else o_carrier_id = r->O_CARRIER_ID;
    load_row(o_table);
}

end_order_load()
{
    table_close(o_table);
}

```

```

table_info ol_table[1];

DBDATETIME ol_delivery_d;

begin_orderline_load()
{
    table_open(ol_table, "order_line");
}

orderline_load(r)
    orderline_row *r;
{
    static orderline_row *old;

    if (r != old)
    {
        old = r;
        bind_ID(ol_table, 1, &r->OL_O_ID);
        bind_ID(ol_table, 2, &r->OL_D_ID);
        bind_ID(ol_table, 3, &r->OL_W_ID);
        bind_ID(ol_table, 4, &r->OL_NUMBER);
        bind_ID(ol_table, 5, &r->OL_I_ID);
        bind_ID(ol_table, 6, &r->OL_SUPPLY_W_ID);
        bind_SYBDATE(ol_table, 7, &ol_delivery_d);
        bind_COUNT(ol_table, 8, &r->OL_QUANTITY);
        bind_MONEY(ol_table, 9, &r->OL_AMOUNT);
        bind_TEXT(ol_table, 10, r->OL_DIST_INFO);
    }

    to_sybase_date(&r->OL_DELIVERY_D, &ol_delivery_d);
    load_row(ol_table);
}

end_orderline_load()
{
    table_close(ol_table);
}

table_info no_table[1];

begin_neworder_load()
{
    table_open(no_table, "new_order");
}

neworder_load(r)
    neworder_row *r;
{
    static neworder_row *old = NULL;

    if (r != old)
    {
        old = r;
        bind_ID(no_table, 1, &r->NO_O_ID);
        bind_ID(no_table, 2, &r->NO_D_ID);
        bind_ID(no_table, 3, &r->NO_W_ID);
    }

    load_row(no_table);
}

end_neworder_load()
{
    table_close(no_table);
}

table_info s_table[1];

begin_stock_load()
{
    table_open(s_table, "stock");
}

stock_load(r)
    stock_row *r;
{
    static stock_row *old = NULL;

    if (r != old)
    {
        old = r;
        bind_ID(s_table, 1, &r->S_I_ID);
        bind_ID(s_table, 2, &r->S_W_ID);
        bind_COUNT(s_table, 3, &r->S_QUANTITY);
        bind_COUNT(s_table, 4, &r->S_YTD);
        bind_COUNT(s_table, 5, &r->S_ORDER_CNT);
        bind_COUNT(s_table, 6, &r->S_REMOTE_CNT);
        bind_TEXT(s_table, 7, r->S_DIST_01);
        bind_TEXT(s_table, 8, r->S_DIST_02);
        bind_TEXT(s_table, 9, r->S_DIST_03);
        bind_TEXT(s_table, 10, r->S_DIST_04);
        bind_TEXT(s_table, 11, r->S_DIST_05);
        bind_TEXT(s_table, 12, r->S_DIST_06);
        bind_TEXT(s_table, 13, r->S_DIST_07);
        bind_TEXT(s_table, 14, r->S_DIST_08);
        bind_TEXT(s_table, 15, r->S_DIST_09);
        bind_TEXT(s_table, 16, r->S_DIST_10);
        bind_TEXT(s_table, 17, r->S_DATA);
    }

    load_row(s_table);
}

end_stock_load()
{
    table_close(s_table);
}

LOGINREC *login;

table_open(table, table_name)
    table_info *table;
    char *table_name;
{
    int message_handler(), error_handler();

    /* save the table name and clear the row count */
    strncpy(table->name, table_name, sizeof(table->name));
    table->count = 0;

    /* make note we have established a connection */
    DBSETLUSER(login, "sa");
    DBSETLAPP(login, table_name);
    BCP_SETL(login, TRUE);
    DBSETLPACKET(login, 4096);

    /* establish a connection with the server specified by DSQUERY env var */
    table->dbproc = dbopen(login, NULL);
    if (table->dbproc == NULL)
        error("Can't establish connection. Is the DSQUERY environment set?\n");

    /* select the database to use */
    if (dbuse(table->dbproc, "tpcc") != SUCCEED)
        error("Can't select database: TPCC\n");

    /* prepare to do a bulk copy */
    if (bcp_init(table->dbproc, table->name, NULL, NULL, DB_IN) != SUCCEED)
        error("Can't initialize the bulk copy to table %s\n", table);
}

```

```

table_close(table)
table_info *table;
{
if (bcp_done(table->dbproc) == -1)
error("Problems completing the bulk copy in %s table.\n", table->name);
dbclose(table->dbproc);
}

bind_int(table, column, addr)
table_info *table;
int column;
int *addr;
{

if (bcp_bind(table->dbproc, addr, 0, -1, NULL, 0, SYBINT4, column)
!= SUCCEED)
error("Can't bind INT to col %d in %s table\n", column, table->name);
}

bind_double(table, column, addr)
table_info *table;
int column;
double *addr;
{
if (bcp_bind(table->dbproc, addr, 0, -1, NULL, 0, SYBFLT8, column)
!= SUCCEED)
error("Can't bind DOUBLE to col %d in %s table\n", column, table->name);
}

bind_str(table, column, addr)
table_info *table;
int column;
char *addr;
{
if (bcp_bind(table->dbproc, addr, 0, -1, "", 1, SYBCHAR, column) != SUCCEED)
error("Can't bind STRING to col %d in %s table\n", column, table->name);
}

bind_SYBDATE(table, column, addr)
table_info *table;
int column;
DBDATETIME *addr;
{
if (bcp_bind(table->dbproc, addr, 0, -1, NULL, 0, SYBDATETIME, column)
!= SUCCEED)
error("Can't bind SYBDATE to col %d in %s table\n", column, table->name);
}

error_handler(dbproc, msgno, msgstate, severity, msgtext, srvname,
procname, line)
/*****
*****
error_handler deals with error messages
*****
*****/
DBPROCESS *dbproc;
DBINT msgno;
int msgstate;
int severity;
char *msgtext;
char *procname;
DBUSMALLINT line;
{
message("Error %d at line %d: %s\n", msgno, line, msgtext);
return INT_CANCEL;
}

message_handler(dbproc, msgno, msgstate, severity, msgtext, srvname,
procname, line)
/*****
*****
message_handler deals with informational messages
*****
*****/
DBPROCESS *dbproc;
DBINT msgno;
int msgstate;
int severity;
char *msgtext;

```

```

char *procname;
DBUSMALLINT line;
{
if (severity != 0)
message("Message from DBLIB: %s\n", msgtext);
}

begin_load()
{

/* start the row counter */
/* use dblib version 10 for numeric datatypes */
dbsetversion(DBVERSION_100);

/* initialize dblib */
if (dbinit() != SUCCEED)
error("Can't initialize the DB library\n");

/* install a message handler */
(void)dbmsghandle(message_handler);
(void)dberrhandle(error_handler);

/* allocate a login record and fill it in */
login = dblogin();
if (login == NULL)
error("Can't allocate a login record.\n");
}

end_load()
{
/* release the login record */
dbloginfree(login);
}

to_sybase_date(date, sybdate)
DATE *date;
DBDATETIME *sybdate;
{
sybdate->dtdays = date->day;
sybdate->dttime = date->sec*300;
}

from_sybase_date(sybdate, date)
DBDATETIME *sybdate;
DATE *date;
{
date->day = sybdate->dtdays;
date->sec = sybdate->dttime / 300;
}

```

## *Appendix C: Tunable Parameters*

---

The HP-UX operating system tunable parameters employed to generate the kernel for the HP 9000 V2250 Enterprise Server and the 12 HP 9000 Model D230 Enterprise Server clients are listed below. Included as well are the Sybase Adaptive Server Enterprise 11.5 and TUXEDO ETP parameters.

---

### **C.3 HP-UX Configuration - Clients**

---

---

---

#### *system parameters for client*

---

---

\* Drivers & Subsystems

c720  
sdisk  
sctl  
\*  
lan2  
vglan0  
mux2  
asio0  
\*  
eisa  
prf  
cdfs  
\*  
clone  
dlpi  
echo  
ffs  
hpstreams  
inet  
inet\_clts  
inet\_cots  
ldterm  
lv  
lvm  
dmem  
dev\_config  
ni  
nm

```

pckt
pipedev
pipemod
ptem
ptm
pts
sad
sc
strlog
target
timod
tirdwr
uipc
tpiso

*****
* Tunables
*****
default_disk_ir 0
nstrpty      60
maxfiles     4096
maxfiles_lim 4096
fs_async     1
bufpages     1024
maxusers     3700
maxuprc      3650
nproc        3700
maxswapchunks 2048
nfile        12000
nflocks      4096
ninode        3000
npty          100
shmmni        16
shmmax        0x7ffffff
shmseg        16
maxssiz      0x8000000
timezone      480
unlockable_mem 1
swapmem_on    0
msgmap        (MSGTQL+2)
msgmax        32768
msgmnb        65535
msgmni        (NPROC)
msgssz        512
msgseg        (MSGTQL*4)
msgtql        (NPROC)
semaem        16384
semmap        (SEMMNI+2)
semmni        (NPROC)
semms         (SEMMNI)
semms         (SEMMNI)
semnu         (SEMMNI)
semume        10
semvmx        32768
nhtbl_scale   2

```

---

## *system parameters for server*

---

```

*****
*****
* Dfile for V-class TPC-C
*****
*****
* Bus-Dependent subsystems
*****
*****
epic
*****
* SCSI drivers
*****
*
sdisk

```

```

stape
setl
c720
*****
*
* Other device drivers
*****
*
btlan6
fddi4
consp1
sppcore
*****
* Pseudo drivers
*****
*
prf
asyncdsk
onyxe
*****
* Subsystems
*****
*
uipc
inet
nms
lvm
tun
*****
* Streams, DLIP, and Streams-based PTY Drivers/Modules
* Note: To remove the Streams PTY driver from the dfile, you need to
* yank out the following items:
* ptm, pts, ldterm, ptem, pckt, and nstrpty 60
*****
*
hpstreams
clone
echo
sad
strlog
timod
tirdwr
sc
pipemod
pipedev
ffs
dlpi
ptm
pts
ldterm
ptem
pckt
*****
*
* Tunables
*****
*
default_disk_ir 0
fs_async        0
nhtbl_scale     1
nstrpty         60
npty            50
STRMSGSZ        65536
maxfiles        2048
maxfiles_lim    2048
nflocks         2048
bufpages        1024
maxusers        64
maxuprc         192
nproc           256
maxswapchunks   10000
nfile           7500
ninode          7500
* Shared Memory
shmseg          8
shmmni          16
shmmax          17179869184
*Misc

```

```
maxssiz      0x10000000
maxdsiz      0x30000000
timezone     480
unlockable_mem 1
swapmem_on   0
```

---

## C.4 Sybase Adaptive Server Enterprise 11.5 Parameters

---

---

### *sybase server parameters*

---

```
#####
#####
#
#           Configuration File for the Sybase SQL Server
#
#           Please read the System Administration Guide (SAG)
#           before changing any of the values in this file.
#
#####
#####
```

[Configuration Options]

[General Information]

[Backup/Recovery]

```
recovery interval in minutes = 32767
print recovery information = DEFAULT
tape retention in days = DEFAULT
```

[Cache Manager]

```
number of oam trips = DEFAULT
number of index trips = DEFAULT
procedure cache percent = 1
memory alignment boundary = DEFAULT
global async prefetch limit = 90
global cache partition number = 16
```

[Named Cache:c\_cust\_index]

```
cache size = 310M
cache status = mixed cache
cache status = HK ignore cache
cache replacement policy = DEFAULT
local cache partition number = 8
```

[2K I/O Buffer Pool]

```
pool size = 310M
wash size = 64K
local async prefetch limit = 0
```

[Named Cache:c\_cust\_non\_index]

```
cache size = 370M
cache status = mixed cache
cache status = HK ignore cache
cache replacement policy = DEFAULT
local cache partition number = 8
```

[2K I/O Buffer Pool]

```
pool size = 370M
wash size = 64K
local async prefetch limit = 0
```

[Named Cache:c\_customer]

```
cache size = 8M
cache status = mixed cache
cache replacement policy = relaxed LRU replacement
local cache partition number = 16
```

[2K I/O Buffer Pool]

```
pool size = 8M
wash size = 256 K
local async prefetch limit = 0
```

[Named Cache:c\_log]

```
cache size = 20M
cache status = log only
cache replacement policy = DEFAULT
local cache partition number = 1
```

[2K I/O Buffer Pool]

```
pool size = 3M
wash size = 512 K
local async prefetch limit = 0
```

[4K I/O Buffer Pool]

```
pool size = 17M
wash size = 7M
local async prefetch limit = 0
```

[Named Cache:c\_no]

```
cache size = 180M
cache status = mixed cache
cache replacement policy = DEFAULT
local cache partition number = 4
```

[2K I/O Buffer Pool]

```
pool size = 180M
wash size = 2M
local async prefetch limit = 0
```

[Named Cache:c\_no\_order\_index]

```
cache size = 50M
cache status = mixed cache
cache status = HK ignore cache
cache replacement policy = relaxed LRU replacement
local cache partition number = 4
```

[2K I/O Buffer Pool]

```
pool size = 50M
wash size = 64 K
local async prefetch limit = 0
```

[Named Cache:c\_ol]

```
cache size = 600M
cache status = mixed cache
cache replacement policy = DEFAULT
local cache partition number = 16
```

[2K I/O Buffer Pool]

```
pool size = 600M
wash size = 2M
local async prefetch limit = 0
```

[Named Cache:c\_ol\_index]

```
cache size = 180M
cache status = mixed cache
cache status = HK ignore cache
cache replacement policy = DEFAULT
local cache partition number = 8
```

[2K I/O Buffer Pool]

```
pool size = 180M
wash size = 1M
local async prefetch limit = 0
```

[Named Cache:c\_orders]

```
cache size = 725M
cache status = mixed cache
cache replacement policy = DEFAULT
local cache partition number = 16
```

[2K I/O Buffer Pool]  
 pool size = 650M  
 wash size = 8M  
 local async prefetch limit = 0

[16K I/O Buffer Pool]  
 pool size = 75M  
 wash size = 4M  
 local async prefetch limit = 0

[Named Cache:c\_stock]  
 cache size = 10370M  
 cache status = mixed cache  
 cache replacement policy = DEFAULT  
 local cache partition number = 16

[2K I/O Buffer Pool]  
 pool size = 10370M  
 wash size = 10M  
 local async prefetch limit = 0

[Named Cache:c\_stock\_index]  
 cache size = 770M  
 cache status = mixed cache  
 cache status = HK ignore cache  
 cache replacement policy = relaxed LRU replacement  
 local cache partition number = 16

[2K I/O Buffer Pool]  
 pool size = 770M  
 wash size = 128 K  
 local async prefetch limit = 0

[Named Cache:c\_wid]  
 cache size = 35M  
 cache status = mixed cache  
 cache status = HK ignore cache  
 cache replacement policy = relaxed LRU replacement  
 local cache partition number = 8

[2K I/O Buffer Pool]  
 pool size = 35M  
 wash size = 64 K  
 local async prefetch limit = 0

[Named Cache:default data cache]  
 cache size = 6M  
 cache status = default data cache  
 cache replacement policy = relaxed LRU replacement  
 local cache partition number = 4

[2K I/O Buffer Pool]  
 pool size = 6M  
 wash size = 512K  
 local async prefetch limit = 90

[Meta-Data Caches]  
 number of open databases = DEFAULT  
 number of open objects = DEFAULT  
 open object spinlock ratio = DEFAULT  
 number of open indexes = DEFAULT  
 open index hash spinlock ratio = DEFAULT  
 open index spinlock ratio = DEFAULT

[Disk I/O]  
 allow sql server async i/o = DEFAULT  
 disk i/o structures = 1024  
 page utilization percent = DEFAULT  
 number of devices = 92  
 disable character set conversions = DEFAULT

[Network Communication]  
 default network packet size = DEFAULT  
 max network packet size = 4096  
 remote server pre-read packets = DEFAULT  
 number of remote connections = DEFAULT  
 allow remote access = DEFAULT  
 number of remote logins = DEFAULT  
 number of remote sites = 2

max number network listeners = 1  
 tcp no delay = DEFAULT  
 allow sendmsg = DEFAULT  
 syb\_sendmsg port number = DEFAULT

[O/S Resources]  
 max async i/os per engine = 512  
 max async i/os per server = 512

[Parallel Query]  
 number of worker processes = DEFAULT  
 memory per worker process = DEFAULT  
 max parallel degree = DEFAULT  
 max scan parallel degree = DEFAULT

[Physical Resources]

[Physical Memory]  
 total memory = 7730000  
 additional network memory = 4956160  
 lock shared memory = 1  
 shared memory starting address = DEFAULT  
 max SQL text monitored = DEFAULT

[Processors]  
 max online engines = 16  
 min online engines = DEFAULT

[SQL Server Administration]  
 default database size = DEFAULT  
 identity burning set factor = DEFAULT  
 allow nested triggers = DEFAULT  
 allow updates to system tables = DEFAULT  
 print deadlock information = DEFAULT  
 default fill factor percent = DEFAULT  
 number of mailboxes = DEFAULT  
 number of messages = DEFAULT  
 number of alarms = DEFAULT  
 number of pre-allocated extents = DEFAULT  
 event buffers per engine = DEFAULT  
 cpu accounting flush interval = 2147483647  
 i/o accounting flush interval = 2147483647  
 sql server clock tick length = DEFAULT  
 runnable process search count = DEFAULT  
 i/o polling process count = DEFAULT  
 time slice = DEFAULT  
 deadlock retries = DEFAULT  
 cpu grace time = DEFAULT  
 number of sort buffers = DEFAULT  
 number of large i/o buffers = DEFAULT  
 size of auto identity column = DEFAULT  
 identity grab size = DEFAULT  
 lock promotion HWM = DEFAULT  
 lock promotion LWM = DEFAULT  
 lock promotion PCT = DEFAULT  
 housekeeper free write percent = 0  
 partition groups = DEFAULT  
 partition spinlock ratio = DEFAULT  
 allow resource limits = DEFAULT  
 number of aux scan descriptors = DEFAULT  
 SQL Perfmon Integration = DEFAULT  
 allow backward scans = DEFAULT

[User Environment]  
 number of user connections = 404  
 stack size = DEFAULT  
 stack guard size = DEFAULT  
 permission cache entries = DEFAULT  
 user log cache size = 4096  
 user log cache spinlock ratio = DEFAULT

[Lock Manager]  
 number of locks = 12000  
 deadlock checking period = DEFAULT  
 freelock transfer block size = DEFAULT  
 max engine freelocks = 50  
 address lock spinlock ratio = 10  
 page lock spinlock ratio = 10  
 table lock spinlock ratio = 1



[Security Related]

```
systemwide password expiration = DEFAULT
audit queue size = DEFAULT
currread change w/ open cursors = DEFAULT
allow procedure grouping = DEFAULT
select on syscomments.text = DEFAULT
auditing = DEFAULT
current audit table = DEFAULT
suspend audit when device full = DEFAULT
max roles enabled per user = DEFAULT
unified login required = DEFAULT
use security services = DEFAULT
msg confidentiality reqd = DEFAULT
msg integrity reqd = DEFAULT
msg replay detection reqd = DEFAULT
msg origin checks reqd = DEFAULT
msg out-of-seq checks reqd = DEFAULT
secure default login = DEFAULT
dump on conditions = DEFAULT
```

[Extended Stored Procedure]

```
esp unload dll = DEFAULT
esp execution priority = DEFAULT
esp execution stacksize = DEFAULT
xp_cmdshell context = DEFAULT
start mail session = DEFAULT
```

[Error Log]

```
event logging = DEFAULT
log audit logon success = DEFAULT
log audit logon failure = DEFAULT
event log computer name = DEFAULT
```

[Rep Agent Thread Administration]

```
enable rep agent threads = DEFAULT
maximum dump conditions = DEFAULT
```

[Component Integration Services]

```
enable cis = DEFAULT
cis connect timeout = DEFAULT
cis bulk insert batch size = DEFAULT
max cis remote connections = DEFAULT
max cis remote servers = DEFAULT
cis packet size = DEFAULT
cis cursor rows = DEFAULT
cis rpc handling = DEFAULT
```

---

---

## *init\_server.sh*

---

---

```
#!/usr/bin/sh -f
```

```
isql -Usa -P <<< EOF
```

```
dbcc tune(des_bind, 5, warehouse)
dbcc tune(des_bind, 5, district)
dbcc tune(des_bind, 5, item)
dbcc tune(des_bind, 5, stock)
dbcc tune(des_bind, 5, order_line)
dbcc tune(des_bind, 5, orders)
dbcc tune(des_bind, 5, new_order)
dbcc tune(des_bind, 5, customer)
dbcc tune(des_bind, 5, history)
```

```
dbcc tune(des_bind, 5, neworder_local)
dbcc tune(des_bind, 5, neworder_remote)
dbcc tune(des_bind, 5, payment_byid)
dbcc tune(des_bind, 5, payment_byname)
dbcc tune(des_bind, 5, order_status_byid)
dbcc tune(des_bind, 5, order_status_byname)
dbcc tune(des_bind, 5, delivery)
dbcc tune(des_bind, 5, stock_level)
```

```
go
```

```
EOF
```

```
isql -Usa -P <<< EOF
```

```
dbcc tune(maxwritedes, 18)
go
dbcc tune(deviochar, -1, "8")
go
dbcc tune("doneinproc", 0)
go
dbcc tune("cleanup", 0)
go
dbcc traceoff(-1)
go
dump tran tpcc with truncate_only
go
EOF
```

```
isql -Usa -P <<< EOF
```

```
use tpcc
go
sp_logiosize "4"
go
EOF
```

---

---

## *cache\_bind.sh*

---

---

```
#!/usr/bin/sh -f
```

```
isql -Usa -P$PASSWORD -e <<< EOF
```

```
use master
go
sp_dboption tpcc, "single user", true
go
```

```
use tpcc
go
checkpoint
go
```

```
/*
** Cache c_log
*/
```

```
sp_bindcache "c_log", "tpcc", "syslogs"
go
```

```
sp_bindcache "c_wid", "tpcc", "sysindexes"
go
sp_bindcache "c_wid", "tpcc", "sysindexes", "sysindexes"
go
```

```
use master
go
sp_dboption tpcc, "single user", false
go
```

```
use tpcc
go
checkpoint
go
```

```
sp_bindcache "c_wid", "tpcc", "item"
go
sp_bindcache "c_wid", "tpcc", "item", "i_clu"
go
sp_bindcache "c_wid", "tpcc", "district"
go
sp_bindcache "c_wid", "tpcc", "district", "d_clu"
```

---

## C.5 Tuxedo UBBconfig

---

```
go
sp_bindcache "c_wid", "tpcc", "warehouse"
go
sp_bindcache "c_wid", "tpcc", "warehouse", "w_clu"
go

/*
** Cache New Order
*/

sp_bindcache "c_no", "tpcc", "new_order"
go
sp_bindcache "c_no_order_index", "tpcc", "new_order", "no_clu"
go

/*
** Cache Order Line
*/

sp_bindcache "c_ol", "tpcc", "order_line"
go
sp_bindcache "c_ol_index", "tpcc", "order_line", "ol_clu"
go

/*
** Cache orders
*/

sp_bindcache "c_orders", "tpcc", "orders"
go
sp_bindcache "c_no_order_index", "tpcc", "orders", "o_clu"
go

/*
** Cache c_stock_index
*/

sp_bindcache "c_stock_index", "tpcc", "stock", "s_clu"
go

/*
** Cache c_stock
*/

sp_bindcache "c_stock", "tpcc", "stock"
go

/*
** Cache c_customer
*/

sp_bindcache "c_customer", "tpcc", "customer"
go

/*
** Cache c_customer_index
*/

sp_bindcache "c_cust_index", "tpcc", "customer", "c_clu"
go

/*
** Cache c_customer_non_index
*/

sp_bindcache "c_cust_non_index", "tpcc", "customer", "c_non1"
go

/*
** Default cache
*/

sp_bindcache "default data cache", "tpcc", "history"
go
EOF
```

```
# This is a UBBconfig for a client1-server configuration.
#
# This UBBconfig requires settings for:
# SERVER_NAME CLIENT_NAME MASTER_NAME SERVER_ADDR
# CLIENT_ADDR NODE_NAMES
# TLISTEN_PORT TBRIDGE_PORT
# In addition, it requires setting the things all UBBconfig.gens need:
# IPCKEY           some decent IPCKEY, should be different for each config
# ROOTDIR
# TUXCONFIG
# APPDIR
# ULOGDIR
#
#-----
*RESOURCES
#-----
    IPCKEY 40001
    PERM 0666
    MASTER client1

    MAXACCESSERS3650# 1024 or more
    MAXGTT 1024
    MAXSERVERS37
    MAXSERVICES170 # MAXSERVERS * #-of-services-each-server + 10( for BBL)
    MODEL SHM
    LDBALY

# During benchmark, don't want to scan too often. In particular, while
# the client1s are stabilizing in virtual memory, we don't want to sanity
# scan; and if we do sanity scan, we want large timeouts, since the BRIDGE
# the BBL, the DBBL, and the client1s aren't getting much CPU time during that
# period. Current settings:
# * scan servers every 5 minutes (maximum allowed by TUXEDO);
# * wait 1 minute for sanity responses (maximum allowed by TUXEDO);
# * scan all the BBLs from DBBL every 30 minutes (want one scan in the
#   audited results);
# * timeout a blocking call after 5 minutes (the maximum).
SCANUNIT60
SANITYSCAN5
DBBLWAIT1
BBLQUERY30
BLOCKTIME5

#
#-----
*MACHINES
#-----
DEFAULT:
    TUXCONFIG="/project/iti/conf/TUXconfig.client1"
    ROOTDIR="/project/iti"
    APPDIR="/project/tpcc/bin"
    ULOGPFX="/tmp/TUXEDO_LOG"

# for debugging, put both into the same log on the same machine
# ULOGPFX="/home/iti/conf/tpcc/ULOG"
# but for a big run, need some space, and want them local to the
# machine rather than across the net.

# Leave TUXCONFIG alone on the MASTER machine; over-ride for each
# other machine?
client1LMID=client1
    TUXCONFIG="/project/iti/conf/TUXconfig.client1"
#-----
*GROUPS
#-----
group1    LMID=client1
          GRPNO=1
group2    LMID=client1
          GRPNO=2
group3    LMID=client1
          GRPNO=3
group4    LMID=client1
```

```

GRPNO=4
group5    LMID=client1
          GRPNO=5
group6    LMID=client1
          GRPNO=6
group7    LMID=client1
          GRPNO=7

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

#-----
*SERVERS
#-----
#
# "-" is application-specific arguments to be passed to server
# "-n" is designed to specify server-id

service SRVGRP=group1
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n1"
  RQADDR=tpcc_1 SRVID=1

service SRVGRP=group1
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n2"
  RQADDR=tpcc_2 SRVID=2

service SRVGRP=group1
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n3"
  RQADDR=tpcc_3 SRVID=3

service SRVGRP=group1
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n4"
  RQADDR=tpcc_4 SRVID=4

service SRVGRP=group1
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n5"
  RQADDR=tpcc_5 SRVID=5

service SRVGRP=group2
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n6"
  RQADDR=tpcc_6 SRVID=6

service SRVGRP=group2
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n7"
  RQADDR=tpcc_7 SRVID=7

service SRVGRP=group2
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n8"
  RQADDR=tpcc_8 SRVID=8

service SRVGRP=group2
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n9"
  RQADDR=tpcc_9 SRVID=9

service SRVGRP=group2
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n10"
  RQADDR=tpcc_10 SRVID=10

service SRVGRP=group3
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n11"
  RQADDR=tpcc_11 SRVID=11

service SRVGRP=group3
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n12"
  RQADDR=tpcc_12 SRVID=12

service SRVGRP=group3
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n13"
  RQADDR=tpcc_13 SRVID=13

```

```

service SRVGRP=group3
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n14"
  RQADDR=tpcc_14 SRVID=14

service SRVGRP=group3
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n15"
  RQADDR=tpcc_15 SRVID=15

service SRVGRP=group4
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n16"
  RQADDR=tpcc_16 SRVID=16

service SRVGRP=group4
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n17"
  RQADDR=tpcc_17 SRVID=17

service SRVGRP=group4
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n18"
  RQADDR=tpcc_18 SRVID=18

service SRVGRP=group4
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n19"
  RQADDR=tpcc_19 SRVID=19

service SRVGRP=group4
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n20"
  RQADDR=tpcc_20 SRVID=20

service SRVGRP=group5
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n21"
  RQADDR=tpcc_21 SRVID=21

service SRVGRP=group5
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n22"
  RQADDR=tpcc_22 SRVID=22

service SRVGRP=group5
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n23"
  RQADDR=tpcc_23 SRVID=23

service SRVGRP=group5
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n24"
  RQADDR=tpcc_24 SRVID=24

service SRVGRP=group5
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n25"
  RQADDR=tpcc_25 SRVID=25

service SRVGRP=group6
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n26"
  RQADDR=tpcc_26 SRVID=26

service SRVGRP=group6
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n27"
  RQADDR=tpcc_27 SRVID=27

service SRVGRP=group6
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n28"
  RQADDR=tpcc_28 SRVID=28

service SRVGRP=group6
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
  DVRY_SVC -- -n29"
  RQADDR=tpcc_29 SRVID=29

service SRVGRP=group6
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s

```

---

```
DVRY_SVC -- -n30"
  RQADDR=tpcc_30 SRVID=30

service SRVGRP=group7
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n31"
  RQADDR=tpcc_31 SRVID=31

service SRVGRP=group7
  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n32"
  RQADDR=tpcc_32 SRVID=32
#-----
*SERVICES
#-----
*ROUTING
#-----
```

# Appendix D: RTE Configuration

---

This appendix lists RTE input parameters and code fragments used to generate each transaction input field, to demonstrate the RTE was configured to generate transaction input data as specified in *Clause 2* of the specification

---

## D.1 RTE Parameters

---

---

### *TESTENV*

---

```
#####
# Environment variables for running TPC-C
#####
setenv DATABASE "sybase" # name of the database used to run the test
                        # can be either "oracle", "sybase", or
                        # "sqlserver"
setenv OPS            0 # Set to 1 if using OPS
setenv NT             0 # Set to 1 if using NT
setenv BATCH_TPCC    0 # Set to 1 for "batch_tpc" with the
                        # RUNME interface, 0 for c/s TPC-C.
setenv TESTROOT      results
setenv RESULTS_NAME  run1 # Directory name of RESULTS (put in root of
                        # Directory name of RESULTS (put in root of
                        # ~tpcc). So actual directory is
                        # ~tpcc/${TESTROOT}

setenv TRANS_TIME    65 # Total time to run the test for (in minutes)
setenv CHPNT_INTERVAL 600 # 600Seconds to wait before forcing a checkpoint
setenv CHPNT_INTERVAL2 1800 # Seconds to wait before forcing the second
                        # checkpoint
                        # For Sybase and Sqlserver, this is the # of
                        # seconds after the first checkpoint til the
                        # 2nd checkpoint.

setenv DB_SIZE       4300 # 4100Database size on SUT (<= size actually built)
                        # value in warehouses

setenv SERVER        "sut" # The SUT (Database Server)
setenv NR_SERVER     "1"
```

```

setenv CLIENT "client" # NOTE: the client name needs to have a
# suffix of 1,2,3,4... etc starting with
# 1 and going to the number of clients. The
# actual client names will be client1,
# client2, client3, etc. You need to put
# the base client name here.
setenv NR_CLIENT "12" # number of clients

setenv DRIVER "driver" # NOTE: the driver name needs to have a
# suffix of 1,2,3,4... etc starting with
# 1 and going to the number of drivers. The
# actual driver names will be driver1,
# driver2, driver3, etc. You need to put
# the base driver name here.
setenv NR_DRIVER "6" # number of drivers
setenv NR_LAN "2" # For multiple lans between driver and client

#
# Audit related stuff + misc
#
setenv CONSISTANCY 0 # run consistancy checks before/after run
# this should be 1 when doing your final
# performance runs.
setenv OUTPUT_LEVEL 3 # minimum level - 3
# maximum level - 1
# need to set to 1 for durability tests

setenv REMOVE_OUTPUT 0 # set to 1 to compress "success" and
# "deliv_results" files after each run
setenv COMPRESS_OUTPUT 0 # set to 1 to compress "success" and
# "deliv_results" files after each run

setenv CLEAR_LOGS 0 # set to 1 to do a dumptrans after the run

setenv CONFIG_FILE ~tpcc/config/sybase.cfg # database configuration file

#####
# The lines below should really not be modified much (if at all)
#####
# For ODBC
# setenv SHLIB_PATH /opt/odbc/drivers:/opt/odbc/lib
setenv TRANS_NUM 1300000000 # Total number of transactions to run

setenv DELIVERY_LOGS logs # Directory name for logfiles

setenv RPT_WINDOW_SIZE 30 # Reporting window size in number of
# RPT_GRANULARITY; for example,
# window size is 10 minutes if
# RPT_GRANULARITY=30 and
RPT_WINDOW_SIZE=20

setenv TRANS_TYPE 0 # 0=all, 1=new-order, 2=payment,
# 3=order_status, 4=delivery, 5=stock_level

#
# For TPC-C rev 3.1 and later the difference between the LOAD value of
# CLAST_CONST_C and the run value needs to be within 65-119 inclusive
# but can't be 96 or 112
#
setenv CLAST_CONST_C 208 # a run-time constant chosen within [0..255]
setenv CID_CONST_C 498 # a run-time constant chosen within [0..1023]
setenv IID_CONST_C 3415 # a run-time constant chosen within [0..8191]

setenv COPY_ENV 1 # 1 = Copy TESTENV to other Drivers.
# 0 = DO NOT copy. It is the tester's
# responsibility to make TESTENVs on all
# the other drivers.

#
# The following emulex communication values are measured, do not change these
#
#setenv COMM_ADJUST_NEWO 0.83 # new-order comm delay
#setenv COMM_ADJUST_PMT 0.35 # payment comm delay
#setenv COMM_ADJUST_ORDS 0.47 # order-status comm delay
#setenv COMM_ADJUST_DVRY 0.29# delivery comm delay

```

```

#setenv COMM_ADJUST_STKL 0.27# stock-level comm delay

#
# The following COMM delays should be used when using the HUB solution
#
setenv COMM_ADJUST_NEWO 0.00 # new-order comm delay, Convert TELNET
to DTCs
setenv COMM_ADJUST_PMT 0.00 # payment comm delay
setenv COMM_ADJUST_ORDS 0.00 # order-status comm delay
setenv COMM_ADJUST_DVRY 0.00# delivery comm delay
setenv COMM_ADJUST_STKL 0.00# stock-level comm delay

#
# The following menu value are measured for Emulex, do not change these
#
#setenv NEWO_MENU 0.56 # new order menu RTE delay
#setenv PMT_MENU 0.41 # payment menu RTE delay
#setenv OS_MENU 0.32 # order status menu RTE delay
#setenv DVRY_MENU 0.45 # delivery menu RTE delay
#setenv STKL_MENU 0.46 # stock menu RTE delay

#
# Use the following menu times if using HUBs instead of Emulex
#
setenv NEWO_MENU 0.00 # new order menu RTE delay
setenv PMT_MENU 0.00 # payment menu RTE delay
setenv OS_MENU 0.00 # order status menu RTE delay
setenv DVRY_MENU 0.00 # delivery menu RTE delay
setenv STKL_MENU 0.00 # stock menu RTE delay

#
# Keying times Don't change these unless doing special tests. They need
# to be float values.
#
setenv NEWO_KEY 18.01 # new order keying time (18.0)
setenv PMT_KEY 3.01 # payment keying time (3.0)
setenv OS_KEY 2.01 # order status key time (2.0)
setenv DVRY_KEY 2.01 # delivery key time (2.0)
setenv STKL_KEY 2.01 # stock level key time (2.0)

#
# Think times. Twiddle these as needed. They need to be float values.
#
setenv NEWO_THINK 12.12 # new order keying time (12.20)
setenv PMT_THINK 12.05 # payment keying time (12.20)
setenv OS_THINK 10.10 # os keying time (10.25)
setenv DVRY_THINK 5.05 # delivery keying time (5.20)
setenv STKL_THINK 5.05 # stock level keying time (5.20)

setenv RANDOMIZE_OUTPUT 1 # Specifies the percentage of users that should
# output full terminal data (the works) even
# if the OUTPUT_LEVEL is not at 1

```

---



---

## get\_tpcc\_env.c

---



---

```

/*****
*****
@(#) Version: A.10.10 $$

(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****/

get_tpcc_env()
/*****
*****/

Get the environment variables
*****/
{
char *epr;

```

```

long i;

eptr = getenv("TPCCWAR");
if (eptr == NULL) no_warehouse = -1;
else no_warehouse = atoi(eptr);

eptr = getenv("TPCCDIS");
if (eptr == NULL) no_dist_pw = DIST_PER_WARE;
else no_dist_pw = atoi(eptr);

eptr = getenv("TPCCCUS");
if (eptr == NULL) no_cust_pd = CUST_PER_DIST;
else no_cust_pd = atoi(eptr);

eptr = getenv("TPCCORD");
if (eptr == NULL) no_ord_pd = ORD_PER_DIST;
else no_ord_pd = atoi(eptr);

eptr = getenv("TPCCITM");
if (eptr == NULL) no_item = MAXITEMS;
else no_item = atoi(eptr);

eptr = getenv("TPCCSEED");
if (eptr == NULL) tpcc_load_seed = LOADSEED;
else tpcc_load_seed = atoi(eptr);

/*
 * Run time values
 */
eptr = getenv("TRANS_TIME");
if (eptr != NULL) max_duration = atof(eptr)*60;

eptr = getenv("TRANS_NUM");
if (eptr != NULL) max_transactions = atoi(eptr);

eptr = getenv("TRANS_TYPE");
if (eptr != NULL) trans_type = atoi(eptr);

eptr = getenv("OUTPUT_LEVEL");
if (eptr != NULL) {
    i = atoi(eptr);
    if (i == 1) { post_transactions = YES; }
}

/*
 * Constants
 */
eptr = getenv("CLAST_CONST_C");
if (eptr != NULL) CLAST_CONST_C = atoi(eptr);

eptr = getenv("CID_CONST_C");
if (eptr != NULL) CID_CONST_C = atoi(eptr);

eptr = getenv("IID_CONST_C");
if (eptr != NULL) IID_CONST_C = atoi(eptr);

/*
 * Keying times
 */
eptr = getenv("NEWO_KEY");
if (eptr != NULL) key[NEWORDER] = atof(eptr);
eptr = getenv("PMT_KEY");
if (eptr != NULL) key[PAYMENT] = atof(eptr);
eptr = getenv("OS_KEY");
if (eptr != NULL) key[ORDSTAT] = atof(eptr);
eptr = getenv("DVRVY_KEY");
if (eptr != NULL) key[DELIVERY] = atof(eptr);
eptr = getenv("STKL_KEY");
if (eptr != NULL) key[STOCKLEV] = atof(eptr);

/*
 * Think time delays
 */
eptr = getenv("NEWO_THINK");
if (eptr != NULL) think[NEWORDER] = atof(eptr);
eptr = getenv("PMT_THINK");
if (eptr != NULL) think[PAYMENT] = atof(eptr);
eptr = getenv("OS_THINK");
if (eptr != NULL) think[ORDSTAT] = atof(eptr);

```

```

eptr = getenv("DVRVY_THINK");
if (eptr != NULL) think[DELIVERY] = atof(eptr);
eptr = getenv("STKL_THINK");
if (eptr != NULL) think[STOCKLEV] = atof(eptr);

/*
 * Menu emulated delays for Emulex if any at all
 */
eptr = getenv("NEWO_MENU");
if (eptr != NULL) m_fudge[NEWORDER] = atof(eptr);
eptr = getenv("PMT_MENU");
if (eptr != NULL) m_fudge[PAYMENT] = atof(eptr);
eptr = getenv("OS_MENU");
if (eptr != NULL) m_fudge[ORDSTAT] = atof(eptr);
eptr = getenv("DVRVY_MENU");
if (eptr != NULL) m_fudge[DELIVERY] = atof(eptr);
eptr = getenv("STKL_MENU");
if (eptr != NULL) m_fudge[STOCKLEV] = atof(eptr);

/*
 * Transmission emulated delays for Emulex if any at all
 */
eptr = getenv("COMM_ADJUST_NEWO");
if (eptr != NULL) t_fudge[NEWORDER] = atof(eptr);
eptr = getenv("COMM_ADJUST_PMT");
if (eptr != NULL) t_fudge[PAYMENT] = atof(eptr);
eptr = getenv("COMM_ADJUST_ORDS");
if (eptr != NULL) t_fudge[ORDSTAT] = atof(eptr);
eptr = getenv("COMM_ADJUST_DVRVY");
if (eptr != NULL) t_fudge[DELIVERY] = atof(eptr);
eptr = getenv("COMM_ADJUST_STKL");
if (eptr != NULL) t_fudge[STOCKLEV] = atof(eptr);
}

```

---

## D.2 Field Value Generation

---



---

### *generate.c*

---

```

/*****
*****
@(#) Version: A.10.10 $Date: 97/02/20 11:45:59 $

(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****/

#include <stdio.h>
#include <values.h>
#include <unistd.h>
#include <time.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <fcntl.h>
#include <signal.h>
#include <math.h>

#include "shm_lookup.h"
#include "random.h"

#include <time.h>

int CLAST_CONST_C = 208;
int CID_CONST_C = 37;
int IID_CONST_C = 75;

```

```

int trans_type = 0; /* type of transaction 0 == all */

extern ID warehouse;
extern ID district;

extern int no_warehouse;
extern int no_item;
extern int no_dist_pw;
extern int no_cust_pd;
extern int no_ord_pd;
extern int no_new_pd;
extern int tpcc_load_seed;

neworder_gen(t)
neworder_trans *t;
{
    int i;

    t->W_ID = warehouse;

    t->D_ID = RandomNumber(1, no_dist_pw);
    t->C_ID = NURandomNumber(1023, 1, no_cust_pd, CID_CONST_C);

    t->O_OL_CNT = RandomNumber(5, 15);

    for (i=0; i<t->O_OL_CNT; i++)
    {
        t->item[i].OL_I_ID = NURandomNumber(8191, 1, no_item, IID_CONST_C);
        t->item[i].OL_SUPPLY_W_ID = RandomWarehouse(warehouse, scale, 1);
        t->item[i].OL_QUANTITY = RandomNumber(1, 10);
    }

    /* 1% of transactions roll back. Give the last order line a bad item */
    if (RandomNumber(1, 100) == 1)
        t->item[t->O_OL_CNT - 1].OL_I_ID = -1;
}

payment_gen(t)
payment_trans *t;
{
    /* home warehouse is fixed */
    t->W_ID = warehouse;

    /* Random district */
    t->D_ID = RandomNumber(1, no_dist_pw);

    /* Customer is from remote warehouse and district 15% of the time */
    t->C_W_ID = RandomWarehouse(warehouse, scale, 15);
    if (t->C_W_ID == t->W_ID)
        t->C_D_ID = t->D_ID;
    else
        t->C_D_ID = RandomNumber(1, no_dist_pw);

    /* by name 60% of the time */
    t->byname = RandomNumber(1, 100) <= 60;
    if (t->byname)
        LastName(NURandomNumber(255, 0, no_cust_pd/3 - 1, CLAST_CONST_C),
                t->C_LAST);
    else
        t->C_ID = NURandomNumber(1023, 1, no_cust_pd, CID_CONST_C);

    /* amount is random from [1.00..5,000.00] */
    t->H_AMOUNT = RandomNumber(100, 500000);
}

ordstat_gen(t)
ordstat_trans *t;
{
    /* home warehouse is fixed */
    t->W_ID = warehouse;

    /* district is randomly selected from warehouse */
    t->D_ID = RandomNumber(1, no_dist_pw);

    /* by name 60% of the time */
    t->byname = RandomNumber(1, 100) <= 60;

    if (t->byname)
        LastName(NURandomNumber(255, 0, no_cust_pd/3 - 1, CLAST_CONST_C),
                t->C_LAST);
    else
        t->C_ID = NURandomNumber(1023, 1, no_cust_pd, CID_CONST_C);

    delivery_gen(t)
    delivery_trans *t;
    {
        t->W_ID = warehouse;
        t->O_CARRIER_ID = RandomNumber(1,10);
    }

    stocklev_gen(t)
    stocklev_trans *t;
    {
        t->W_ID = warehouse;
        t->D_ID = district;
        t->threshold = RandomNumber(10, 20);
    }

    int get_trans_type()
    /******
    * get_trans_type selects a transaction according to the weighted average
    * For TPC-C rev 3.0 and less and TPC-C rev 3.2 this is:
    *   new-order : ???
    *   payment   : 43.0%
    *   order stat: 4.0%
    *   delivery  : 4.0%
    *   stock     : 4.0%
    *****/
    /******
    {
        static double weight[] = { 0.0, 0.0, .4305, .0405, .0405, .0405 };
        double drand48();
        int type;
        double r;

        /* choose a random number between 0.0 and 1.0 */
        if (trans_type == 0) {
            r = drand48();

            /*
            * select one of STOCKLEV, DELIVERY, ORDSTAT and PAYMENT
            * based on weight
            */
            for (type = STOCKLEV; type > NEWORDER; type--) {
                r -= weight[type];
                if (r < 0) break;
            }
        } else {
            /* user wants only a certain type (say all stocklevel) so do that
            instead */
            type = trans_type;
        }
        /* return the value of the selected card, or NEWORDER if none selected */
        return type;
    }
}

```



## *Appendix E: Disk Storage*

---

The calculations used to determine the storage requirements for the 8 hour logical log and the 180-day space are contained in this appendix.

**Table E.1: Data Space Calculations: Static + 180 Day Growth**

Note : Numbers are in KBytes unless otherwise specified						
Warehouses	4300	tpmC	52,117.80	tpmC/W	12.12	
Table	Rows	Data	Index	5% Space	8H Space	Total Space
Warehouse	4,300	8,602	34	432		9,068
District	43,000	8,652	40	435		9,127
Item	100,000	9,572	48	192		9,812
New-order	38,700,000	422,952	5,094		86,000	514,046
History	129,000,000	7,056,736	0		1,368,490	8,425,226
Orders	129,000,000	3,486,488	42,004		684,269	4,212,761
Customer	129,000,000	86,000,000	6,671,954	1,853,439		94,525,393
Order-line	1,290,120,893	78,189,146	1,028,800		15,362,475	94,580,421
Stock	430,000,000	143,333,334	791,902	2,882,505		147,007,741
Totals		318,515,482	8,539,876	4,737,003	17,501,233	349,293,594
Segment	LogDev Cnt.	Seg. Size	Needed	Overhead		Not Needed
wdino	1	614,400	547,473	5,475		61,452
history	2	8,806,400	8,509,478	85,095		211,827
orders	12	4,556,800	4,254,888	42,549		259,363
customer	13 + 8 index	107,216,896	95,470,647	954,706		11,746,249
order_line	12	96,921,600	95,526,225	955,262		440,113
stock	40	164,797,440	148,477,818	1,484,778		14,834,844
master	1	277,504	0	0		277,504
tmpdb	1	61,140	0	611		60,529
Totals		383,252,180	352,786,530	3,528,477		27,891,880
Dynamic space	85,981,667	Sum of Data for Order, Order-Line and History (excluding free extents)				
Static space	249,339,171	Data + Index + 5% Space + Overhead - Dynamic space				
Free space	20,039,462	Total Seg. Size - Dynamic Space - Static Space - Not Needed				
Daily growth	16,674,141	(Dynamic space/W * 62.5)* tpmC				
Daily spread	(4,971,749)	Free space - 1.5 * Daily growth (zero if negative)				
180 day (KB)	3,250,684,488	Static space + 180 (daily growth + daily spread)				
180 day (GB)	3100.09	GB—Excludes OS, Paging and RDBMS Logs				
Log per N-O txn	1.824	Number of 2K blocks per New-Order transaction				
8 Hour Log (GB)	87.03	GB				
OS+SWAP (GB)	20.00	GB				
Total Space Needed	3207.13	GB				
Priced-System Configuration			Size (MB)	Quantity		Total ( MB)
40GB-Icicle (MB)			39,598	27		1,069,146
90GB-Icicle (MB)			86,186	26		2,240,836
Total Storage in Priced System (GB)						3,232 GB

## *Appendix F: Price Quotes*

---

The following pages contain the price quotes for the hardware included in this FDR.



# SUN DATA

To: Sunny Senyel  
 Hewlett-Packard Company  
 18111 Pruneridge Avenue  
 Cupertino, CA 95014

From: Sandy Potter  
 Sun Data Corporation  
 1 Sun Court  
 Norcross, GA 30092.

Date: February 4, 1998

Item Description	Part No.	Unit Price	Qty.	Extended price	5 Year Service
HP 9000 V2250 Enterprise Server	A5081A	89,080	1	89,080	100,944
Add'l 240 MHz PA8200 CPUic	A5005A Opt. 0D1	24,080	10	305,440	78,588
System Mgmt. Station	A4802A	8,059	1	8,059	
Memory carrier	A3780A	6,570	4	26,280	
128 MB DIMM for a total of 1 GB	Option UTC	11,680	18	180,880	
6x6 Slot PCI Card Cage	A3728A Opt. 0D1	4,380	3	13,140	
PCI FWD SCSI-2 Card	A4800A Opt. 0D1	876	16	14,016	
PCI Dual FDDI LAN Adapter	A3738A Opt. 0D1	2,756	1	2,756	
PCI FDDI LAN Adapter SAW & Lic.	J8626AA	292	1	292	
4 GB DDS-2 DAT Drive	A3183A Opt. 0DZ	1,205	1	1,205	
HP-UX 11.0 Sys Media, CD-ROM	B3820EA, Opt.AAF	380	1	380	
PCI 64-port MUX Card	J3583A Opt. 0D1	1,091	1	1,091	
16 port RS232 DB25 port module	J2485A	642	1	642	
25 ft cable	J3685A	110	1	110	
5.5 kVA UPS	A3588A Opt. 032	7,300	1	7,300	2,700
1.3 kVA HP UPS Rackmount	A2988A Opt. 032	1,825	14	26,550	2,418
1.6m Field Integrated Cabinet	C2788A	1,318	14	18,452	
Auto Raid Array Model 12H	A3700 AZ	3,504	59	185,712	254,287
Two 88 MB Controllers w/ Auto Raid	Opt. 203	11,872	63	634,518	
Twelve 4.3 GB disk modules	Opt. 112	12,814	27	340,578	
Twelve 8.1 GB disk modules	Opt. 132	18,710	26	512,460	
0.8m 68-pin high density male connec.	Option 801	88	37	3,256	
2.6m HDTS88 to HCT888 Connec.	C2924A	102	19	1,632	
HP 9000 Model D230	A3574A	8,159	12	74,028	29,102
1 MB cache	A3488A Opt. 0D1	1,480	12	17,520	
256 MB high density memory	A3584A Opt. 0D1	4,234	36	152,424	
Console	C1084GX	393	12	4,716	1,256
4 GB internal disk	A3843A Opt. 0D1	1,132	12	13,584	7,517
Comb. 10base/1100 VG adap. card	A3402A Opt. 0D1	583	48	27,984	
HP C/ANSI C Compiler	B3901BA Opt. AH0	1,150	1	1,150	1,757
HP Advantstack 2000 10Base-T mod	J3102A	912	3	2,736	2,334
HP Advantstack 2000 FDDI mod.	J3108A	4,014	1	4,014	2,333
HP Advantstack Switch 2000	J3100B	1,824	1	1,824	1,555

Note: Above configurations are generally available and prices are guaranteed for 90 days. The service prices are based on a service agreement for the entire system. The above prices include a purchase discount which may vary if purchased separately. Pricing is subject to approved credit terms.

Sun Data, Inc. • One Sun Court • P.O. Box 926020 • Norcross, GA 30092  
 (770) 449-6116 • (888) SUN-DATA • Fax: (770) 448-7726  
<http://www.sundata.com>





Sunny Sanyal  
Hewlett Packard  
19111 Pruneridge Avenue  
Cupertino, CA 95014

February 5, 1998

Dear Sunny,

Below is the Sybase pricing for Sybase Adaptive Server. Please use this for the joint HP/Sybase TPC-C pricing for the various quotes. All TPC-C results that include pricing must be reviewed and approved by our HP Strategic Alliance Manager, Russ Puryear, prior to publication.

Support Pricing is 16% of list (no discounts).

High-End Pricing (machine has physical capacity of greater than eight processors)

ASE Deployment Pricing	\$295,000 (this pricing has no development attached to it)
Development Pricing	\$9,600 (8 user license)
Open Client	<u>\$795</u>
Total Pdt Pricing	\$305,395
Less 10% Discount	<u>(\$30,540)</u>
Total Net Pricing	\$274,855
Support Pricing	
16% x 305,295 x 5 yrs	<u>\$244,316</u>
Total Price	<u>\$519,171</u>

Regards,

Andrea L. Louks  
Pricing Operations Manager  
Sybase, Inc.

cc: Mike Regan  
Russ Puryear  
Lee Abramson

Sybase, Inc.  
6475 Christie Avenue  
Emeryville, CA 94608  
(510) 922-3500  
Fax (510) 922-0178  
www.sybase.com



February 6, 1998

Hewlett Packard Company

Subject: Pricing for Tuxedo 4.2.2

Dear Mr. Sanyal ;

At present BEA Tuxedo version 4.2.2 is available for customer purchase, although by September 30, 1998 it will be removed from general availability. It is replaced by Tuxedo 6.x from BEA Systems, Inc. and as of January, 1998 version 6.3 has both static and dynamic libraries available. Please contact BEA's product manger for Tuxedo to get more details on this version.

Pricing for release 4.2.2 is as follows, with HP machine classifications determined using the table following the price summary:

**TUXEDO 4.2.x Packaging/Pricing - includes /T, /Q and /C features in one bundle**

Machine Class	List price	Maintenance (5 x 8) per year	Maintenance (7 x 24) per year
Class 1	\$1,440.00	\$216.00	\$316.80
Class 2	\$4,200.00	\$630.00	\$924.00
Class 3	\$9,000.00	\$1,350.00	\$1,980.00
Class 4	\$18,000.00	\$2,700.00	\$3,960.00
Class 5	\$26,400.00	\$3,960.00	\$5,808.00
Class 6	\$58,800.00	\$8,820.00	\$12,936.00
Class 7	\$94,800.00	\$14,220.00	\$20,856.00

PS For purposes of future TPC activity we recommend the use of Tuxedo 6.3 available through our Core Functionality Services (CFS) program. This is a replacement for version 4.2.2 or earlier. This should be denoted as Tuxedo 6.3 CFS in all publications. It is priced as shown below.

### Tuxedo Core Functionality Services (CFS) Program Product Pricing and Description

TUX CFS is a limited function product from BEA that offers a subset of TUXEDO 6.3 capabilities. The following standard 6.X features ARE NOT AVAILABLE WITH TUX-CFS AND CANNOT BE SUPPORTED BY TUX-Core:

1. /WS
2. /Q
3. /COBOL
4. /DCE
5. Transaction
6. Events
7. DOMAINS
8. Admin API
9. Admin GUI
10. ACLs (Access Control List security option)

TUX-CFS provides a basic level of middleware support for distributed computing, and is best used by organizations with substantial resources and knowledge for advanced distributed computing implementations. TUX-CFS prices are server only and are based on the overall performance characteristics of the server and uses the same five tier computer classification as TUXEDO 6.1. Prices range from \$3,000.00 for Tier 1 to \$250,000 for Tier 5. Under this pricing option EVERY server running TUX-CFS at the user site must have a TUXEDO license installed and pay the appropriate per server license fees.

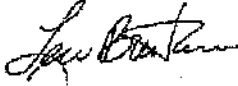
#### BEA Tux/CFS Unlimited User License Fees Per Server

Unlimited User License fees per server	Number of Users	Dollar Amount	Maintenance (5 x 8) per year	Maintenance (7 x 24) per year
Tier 1 -- PC Servers with 1 or 2 CPUs, entry level RISC Uni-processor workstations and servers	Unlimited	\$3,000.00	\$450.00	\$660.00
Tier 2 - PC Servers with 3 or 4 CPUs, Midrange RISC Uni-processor servers and workstations (class 3)	Unlimited	\$12,000.00	\$1,800.00	\$2,640.00
Tier 3 - Midrange Multiprocessors, up to 8 CPUs per system capacity	Unlimited	\$30,000.00	\$4,500.00	\$6,600.00
Tier 4 - Large (more than 8, less than 32 CPUs) and Mainframe Systems (Class 6)	Unlimited	\$100,000.00	\$15,000.00	\$22,000.00
Tier 5 - Massively Parallel Systems, > 32 processors	Unlimited	\$250,000.00	\$37,500.00	\$55,000.00

**Hewlett Packard System Classification for HP-UX systems:**

Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
Uni-processor Workstations	9000/E25 9000/E35 9000/E45 9000/E55 9000/G30 9000/G40	9000/G50 9000/G60 Multi-Processor Workstations 9000/D200, 210, 220, 230	9000/H20 9000/H30 9000/H40 9000/H50 9000/I30 9000/I40 9000/K1XX 9000/D310, 320, 330 9000/D250/60	9000/I50, 60 9000/H60 9000/G70 9000/H70 9000/I70 9000/K2XX 9000/K3XX 9000/K4XX 9000/K5XX 9000/D350/60/70	9000/T50 0 1-16 CPUs	9000/V series all models

Sincerely,



Lew Brentano  
Director, Market Planning





NETLUX

14180 Live Oak Ave., Unit B  
Baldwin Park, Ca. 91760

1-800-730-1730  
Phone #818-851-9737  
Fax #818-851-9837

February 3, 1998

Hewlett Packard  
Sunny Sanyal  
408-447-5604  
408-447-5730

### Quotation

Quantity	Part No.	Description	Unit Price	Total
1	NX-H9	9-port 100Mbps Ethernet Hub	\$47.00	\$ 47.00

Terms and Conditions:  
FOB Origin  
5 Year Warranty  
Prices good for 60 Days

Sincerely,  
Martin Parry  
NETLUX

