Silicon Graphics Computer Systems
Informix Software, Inc.

Silicon Graphics Origin2000 Server
Using IRIX 6.4 S2MP and
INFORMIX-OnLine Dynamic Server 7.3.UC1
Client/Server Configuration

TPC Benchmark™ C
Full Disclosure Re

First Edition
Submitted for review
30 April 1997

First Edition - 30 April 1997

The benchmark results contained in this document were submitted for compliance with version 3.3 of the TPC Benchmark C Standard Specification. The result of that action is to place those benchmark results into the sixty day “under review” status as of 30 April 1997.

Silicon Graphics and Informix Software, Inc. believe that the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. Silicon Graphics and Informix Software, Inc. assume no responsibility for any errors that may appear in this document.

The pricing information in this document is believed to accurately reflect the current prices as of the publication date. However, Silicon Graphics and Informix Software, Inc. provide no warranty on the pricing information in this document.

Benchmark results are highly dependent upon workload, specific application requirements, and systems design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC Benchmark 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, and therefore results obtained in other operating environments may vary significantly. Silicon Graphics and Informix Software, Inc. do 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.

Printed in USA April 1997

Silicon Graphics Part Number:
Informix Part Number:

Origin2000, Challenge S, and S2MP are registered trademarks of Silicon Graphics.

INFORMIX-OnLine and ESQL/C are registered trademarks of Informix Software, Inc.

TPC Benchmark, TPC-C, and tpmC are trademarks of the Transaction Processing Performance Council.

All other trademarks mentioned herein are the property of their respective owners.

Abstract

Overview

This report documents the methodology and results of the TPC
C) conducted on the Silicon Graphics Origin2000 Server using
Dynamic Server version 7.3.UC1 and the BEA Tuxedo 6.1 CFS tra
The operating system used for the benchmark was IRIX 6.4 S2M
was written in C, preprocessed by INFORMIX-ESQL/C and compi
IRIX C compiler.

TPC Benchmark Metrics

The standard TPC Benchmark C metrics, tpmC (transaction per
tpmC (five year capital cost per measured tpmC), and the avai
as required by the benchmark specification.

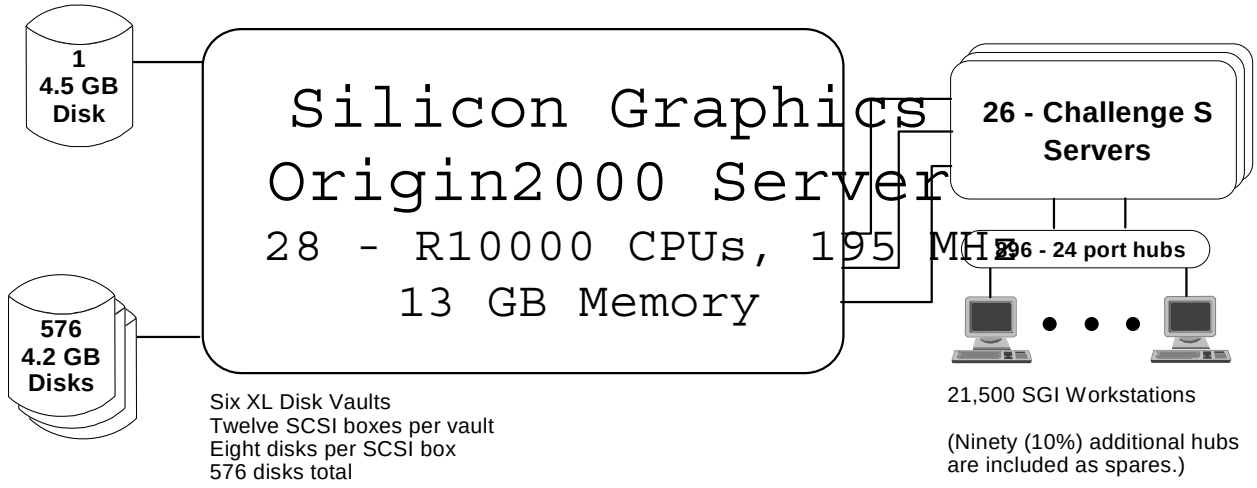
Executive Summary

Pages iv-v contain the executive summary of the benchmark r
Graphics Origin2000 Server using INFORMIX-OnLine Dynamic S
7.3.UC1.

Auditor

The benchmark configuration, environment, and methodology us
validate the test results, along with the pricing model use
tpmC, were audited by Richard Gimarc of Performance Metri
compliance with the relevant TPC specifications.

Silicon Graphics Informix Software	Silicon Graphics Origin2000 Server C/S Configuration - 26 Challenge S front-er			TPC-C Rev 3.1 Report Date: 30 April 19
Total System Cost	TPC-C Throughput		Price/Performance	Availability Date
$3,519,011.50	25,309.20 tpmC		$139.04 per tpmC	29 Oct 9
Processors	Database Manager	Operating System	Other Software	Number of Users
28 R10000 195 MHz	INFORMIX-OnLine Dynamic Server version 7.3.UC1	IRIX 6.4 S2MP	BEA Tuxedo 6.1 CFS	21,500



Six XL Disk Vaults
Twelve SCSI boxes per vault
Eight disks per SCSI box
576 disks total

21,500 SGI Workstations

(Ninety (10%) additional hubs
are included as spares.)

System Components	Server		Client(s)	
	Quantity	Type	Quantity	Type
Processors	28	R10000 195 MHz	1	R4400 200 MHz per
Cache Memory		32 KB Instruction/Data each 4 MB Secondary		16 KB Instructi 1 MB Secondary
Memory		13 GB		256 MB per client
Disk Adapters	18	SCSI Adapters	1	SCSI adapter per
	72	SCSI bus		
Disk Drives	576	4.2 GB SCSI	1	2.0 GB per client
	1	4.5 GB SCSI		
Total Storage		2,423.7 GB		
Transfer Media	1	CD-ROM		
Terminals	1	System Console	1	System Console per
			21,500	SGI Workstations tota
Connectivity	1	4-port FastEthernet Adapter	986	24-port hubs (includes 10% spares)

Silicon Graphics	Silicon Graphics Origin2000 Server	TPC-C Rev 3.3
Informix Software	Client/Server Configuration	Report Date: 30 April 1

Part Number	Description	Quantity	Unit Price	Extended Price	Five Year Maintenance
R1-H64002-4	Origin2000 Rack (2 CPUs) 64 MB memory, CD-ROM, IRIX for 2 CPUs	1	70,000.00	70,000.00	
HU1-RL6402-4	Origin2000 Expansion Rack (2 CPUs) 64 MB memory, IRIX for 2 CPUs	1	59,500.00	59,500.00	
HU1-ML6402-4	Origin2000 Insert Module (2 CPUs/module) 64 MB memory, IRIX for 2 CPUs	4	25,500.00	91,000.00	
HU1-H6402-4	Dual R10000 CPUs	10	39,900.00	399,000.00	
OPN-64U1GB-8	Base Memory - 1 GB	12	53,760.00	645,120.00	
H4-N128	Additional Memory - 128 MB	6	7,168.00	43,008.00	
OPN-64U128-1	Additional Memory - 128 MB	2	3,584.00	7,168.00	
CR-64U1GB-8	Standard memory program	12	(13,440.00)	(161,280.00)	
CR-N128	Standard memory program	6	(1,792.00)	(10,752.00)	
CR-64U128-1	Standard memory program	2	(896.00)	(1,792.00)	
XT-SCSI-4P	4-port Ultra SCSI XIO Card	18	2,100.00	37,800.00	
XT-FE-4TX-6A	XIO 4-port Fast Ethernet Adapter	1	4,200.00	4,200.00	
HU-IC-XP-32	Upgrade Xpress Links - 32 CPU config	1	7,000.00	7,000.00	
HU-IC-17TO32	Upgrade CrayLink cables for 17-32 CPUs	1	10,500.00	10,500.00	
DK-LR2-003	Rack system and expansion destination kit	1	0.00	0.00	
P-S-2TB	6 Full Vault XL (576 disks, 4.2 GB)	88	71,880.00	887,880.00	
FTO-4GB	4.5 GB SE Ultra SCSI system disk	1	0.00	0.00	
WY-55	Wyse 55 terminal (system console)	1	589.00	589.00	
FC-EW5	Full Care Extended Warranty - 5 years	1			245,294.00
	Origin2000 Subtotal (pricing from 1DIRECT)			2,088,941.00	245,294.00
CH-S200-2G64N	Challenge S	26	11,130.00	289,380.00	
CR-CHS-SC4PC	Standard credit for Challenge S	26	(4,340.00)	(112,840.00)	
HU-M128S	128 MB memory upgrade	26	2,800.00	72,800.00	
HU-M64A	64 MB memory upgrade	26	1,400.00	36,400.00	
WY-55	Wyse 55 terminal (system console)	26	589.00	15,314.00	
FC-EW5	Full Care Extended Warranty - 5 years				497.50 4,
	Challenge S Subtotal (pricing from 1DIRECT)			301,054.00	4,497.50
	Hardware Subtotal:			2,389,995.00	249,791.50
AL-HUB24L	Acculan 24 port HUB-Ultra	986	212.50	209,525.00	
SC4-IRIX64-6.4	IRIX 6.4 S2MP operating system	1	0	0.00	
	INFORMIX-OnLine Dynamic Server 7.3	1	310,000.00	310,000.00	223,200.00
	BEA Tuxedo 6.1 CFS	26	3,000	78,000.00	58,500.00
	Software Subtotal:			388,000.00	281,700.00
	Total Product and Maintenance Cost:			2,987,520.50	531,491.50
	Total Five Year Configuration Cost:				$ 3,519,011.50
	tpmC				25,309.20
	Price (U.S. dollars) per tpmC:				$ 139.04

Notes: Audited by Richard Gimarc of Performance Metrics, Inc.

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

Numerical Quantities Summary

Silicon Graphics Origin2000 Server

INFORMIX-OnLine Dynamic Server version 7.3 UC1

MQTh, Computed Maximum Qualified Throughput: 25,309.20

Transaction Mix
(% of Total Transactions)

New-Order	44.66%
Payment	43.17%
Order-Status	4.06%
Delivery	4.05%
Stock-Level	4.06%

Response Time (seconds)

Transaction	Average	90th Percentile	Maximum
New-Order	1.62	1.70	52.19
Payment	1.38	1.94	62.91
Order-Status	0.91	0.95	41.10
Delivery (interactive)	0.32	0.42	0.74
Delivery (deferred)	16.38	47.01	144.00
Stock-Level	1.57	3.06	44.55
Menu:			
New Order	0.34	0.41	20.09
Payment	0.34	0.41	5.84
Order Status	0.34	0.41	2.95
Delivery	0.34	0.41	0.72
Stock Level	0.34	0.41	2.28

Keying/Think Times (seconds)

	Minimum	Average	Maximum
New-Order	18.00 / 0.10	18.00 / 12.05	18.00 /120.00
Payment	3.00 / 0.10	3.00 / 12.03	3.00 / 120.00
Order-Status	2.00/ 0.10	2.00 / 10.06	2.00 / 100.00
Delivery (interactive)	2.00 / 0.10	2.00 / 5.06	2.00 / 50.00
Stock-Level	2.00 / 0.10	2.00 / 5.05	2.00 / 50.00

Test Duration:	
Ramp-up time	19 minutes
Measurement interval (MI)	30 minutes
Transactions completed during MI	1,700,064
Ramp-down time	7 minutes
Checkpointing:	
Number of checkpoints	1
Checkpoint interval	30 minutes
Reproducibility:	
Reported measurement	25,309.20
Reproducibility measurement	25,211.50
Difference	0.39%

Table of Conten

APPENDIXES

FIGURES

TABLES

Preface

Document Structure

The TPC Benchmark C Standard Specification requires test s
submit to the TPC, and make available to the public, a full
result to considered compliant with the specification. The
full disclosure report are specified in Clause 8.

This report is submitted to satisfy the specification s requ
It documents the compliance of the benchmark implementati
reported for the Silicon Graphics Origin2000 Server using
Dynamic Server 7.3.UC1.

In the specification, the main headings in Clause 8 are keye
headings in this report use the same sequence, so that they
or subjects referred to in Clause 8.

Each section in this report begins with the text of the c
Clause 8 of the specification, printed in italic type. The fo
how this benchmark complies with that specific portion of
section where Clause 8 requires extensive listings, the app
report is referenced.

TPC Benchmark C Overview

TPC Benchmark C was developed by the Transaction Processin
Council (TPC). It is the intent of the TPC to develop a s
measure the performance of computer systems executing a wide
tions. Silicon Graphics and Informix Software are active pa
define and develop such a suite of benchmarks.

TPC Benchmark C is an On-Line Transaction Processing (OLTP)
a mixture of read-only and update intensive transactions tha
found in complex OLTP application environments. It does so
breadth of system components associated with such environ
characterized by:

- The simultaneous execution of multiple transactions typ
  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
  keys.
- Databases consisting of many tables with a wide variety o
  and relationships.
- Contention of data access and update.

The performance metric reported by TPC-C is a business thro
the number of orders processed per minute. Multiple transa
simulate the business activity of processing an order, and ea
to a response time constraint. The performance metric for
expressed in transaction-per-minute-C (tpmC). To be complai
standard, all references to tpmC results must include the tp
price-per-tpmC, and the availability date of the priced con

Despite the fact that this benchmark offers a rich environme
OLTP applications, this benchmark does not reflect the ent
requirements. In addition, the extent to which a customer c
reported by a vendor is highly dependent on how closely TPC-
customer application. The relative performance of systems
benchmark does not necessarily hold for other workloads
Extrapolations to other environments are not recommended.

General Items

0.1 Application Code and Definition Statements

The application program (as defined in Clause 2.1.7) must
limited to, the code implementing the five transactions ar

Appendix A contains the TPC-C application code used in this

0.2 Test Sponsor

A statement identifying the benchmark sponsor(s) and other p

Silicon Graphics and Informix Software, Inc. were co-spc
Benchmark C. The benchmark was conducted at Silicon Graphics
mance facility.

0.3 Parameter Settings

Settings must be provided for all customer-tunable paramete
from the defaults found in actual products, including but

- Database tuning options.
- Recover/commit options.
- Consistency/locking options.
- Operating system and application configuration parameter
- Compilation and linkage options and run-time optimizat
  applications, OS, and/or databases.

This requirement can be satisfied by providing a full list

Appendix C contains the IRIX 6.4 S2MP operating system paramet
the kernel for the configuration used in this benchmark. Al
INFORMIX-OnLine database parameters and the BEA Tuxedo 6.1 C
monitor parameters used.

0.4 Configuration Diagrams

Diagrams of both measured and priced configurations must be p
of the differences.

The System Under Test (SUT), a Silicon Graphics Origin2000 s
Figure 0.1, consisted of:

- Twenty-eight R1000 195 MHz CPUs
- 13 GB of memory
- 18 SCSI Adapters with 72 SCSI busses
- 419 SCSI disks at 4.202 GB each
- Two SCSI disks at 4.238 GB each
- One CD-ROM
- One FastEthernet Network Adapter with four ports

The benchmark configuration used Performix s Empower that exe
driver systems to emulate TPC-C users. The benchmark configu
in Figure 0.1. The emulated users on the driver systems were
the client systems under test via local area network (LAN) ar
TCP/IP.

The priced configuration for the Silicon Graphics Origin2000
In that priced configuration, RTE systems, shown in Figure 0.
Acculan 24-port hubs and 21,500 end user devices. Ninety a
included in the priced configuration to serve as spares in

The benchmark configuration contained 421 disks, each with a
The priced configuration includes an additional disks of the
the 180-day storage requirement.

Figure 0.1: Silicon Graphics Origin2000 Server Benchmark Configuration

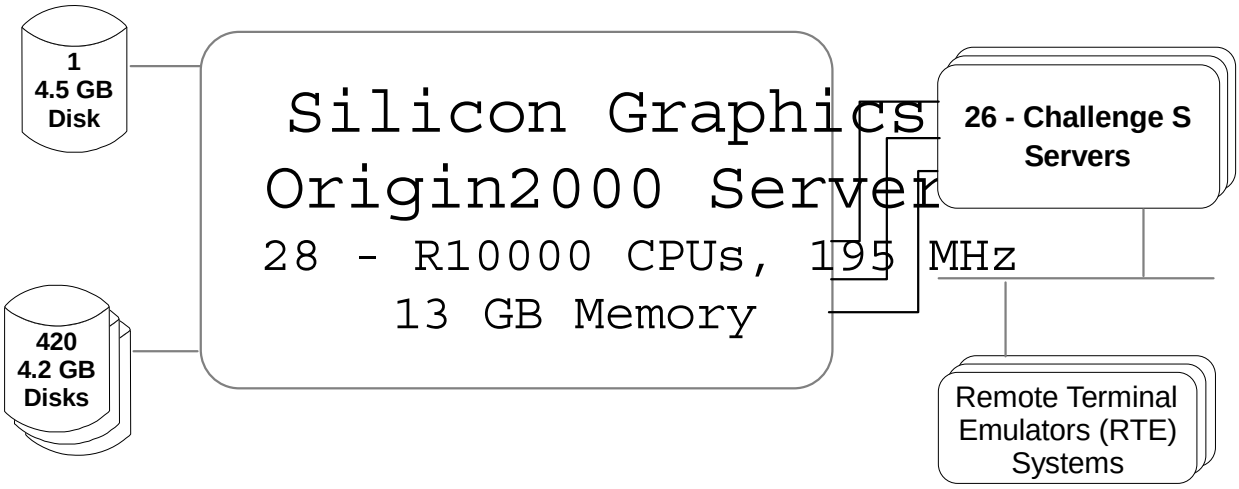
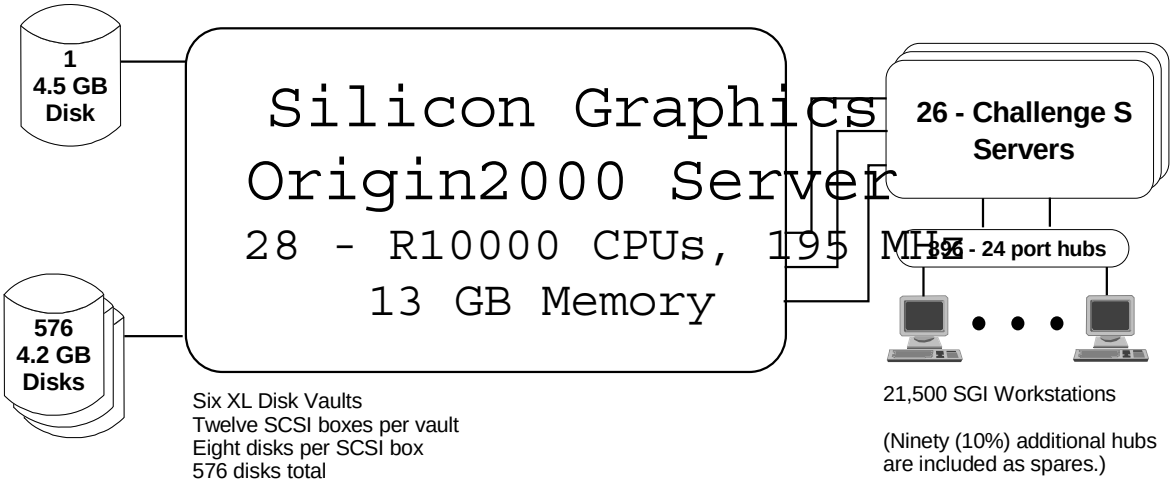


Figure 0.2: Silicon Graphics Origin2000 Server Priced Configuration



Clause 1 Related It

1.1 Table Definitions

Listing must be provided for all table definition statemen
the database.

Appendix B describes the programs that define, create, and
database for TPC-C benchmarks.

1.2 Physical Organization of the Database

The physical organization of tables and indices, within th

Disk space was allocated to INFORMIX-OnLine according to th
4.2. No attempt was made to alter how INFORMIX-OnLine chos
organization of the tables and indices. Indices were define
so that INFORMIX-OnLine loaded the index pages as the datab
in Appendix B populated the tables.

1.3 Insert and Delete Operations

It must be ascertained that insert and/or delete operations
with the TPC-C transaction mix. Furthermore, any restricti
tation that precludes inserts beyond the limits defined i
includes the maximum number of rows that can be inserted a
new roses.

There were no restrictions on insert and delete operations

1.4 Partitioning

While there are a few restrictions placed upon horizontal or
in the TPC-C benchmark, any such partitioning must be discl
must be disclosed. Additional and/or duplicate attributes ir
a statement on the impact on performance.

Partitioning, replication, and additional or duplicated attr
implementation.

Clause 2 Related It

2.1 Random Number Generation

The method of verification for the random number generatic

The RTE software, Empower from Performix, Inc., generated th
used in this benchmark. Empower computes random integers usi
tion of a linear congruential sequence as described in
Programming, Volume2/Seminumerical Algorithms by Donald E. K
1981 by Addison-Wesley Publishing Company.

The seeds for each user were captured and verified by the a
contents of the database were systematically searched, and r
auditor for patterns that would indicated the random number
any type of discernible pattern; none was found.

2.2 Input/Output Screen Layout

The actual layout of the terminal input/output screens mus

The screen layouts corresponded exactly to those in Clauses
and 2.8.3 of the TPC Benchmark C Standard Specification.

2.3 Priced Terminal Feature Verification

The method used to verify that the emulated terminals prov
2.2.2.4 must be explained.

This was verified by the auditor by a direct experiment duri

2.4 Presentation Manager or Intelligent Terminal

Any usage of presentation managers or intelligent terminals

Application code running on the client implemented the TPC-(listing of this code is included in Appendix A. Capabilities c user display/keyboard device, beyond basic ASCII entry and di to cursor positioning. A presentation manager was not used.

2.5 Transaction Statistics

Table 3.1 lists the numerical quantities that Clauses 8.1.3.

Table 2.1: Transaction Statistics

Statistic		Value
New Order	Home warehouse	99.05%
	Remote warehouse	0.95%
	Rolled Back transactions	1.00%
	Average items per order	10
Payment	Home warehouse	85.03%
	Remote warehouse	14.97%
	Non-primary key access	60.09%
Order Status	Non-primary key access	60.25%
Delivery	Skipped Transactions	0
Transaction Mix	New Order	44.66%
	Payment	43.17%
	Order Status	4.06%
	Delivery	4.05%
	Stock Level	4.06%

2.6 Queuing Mechanism

The queuing mechanism used to defer the execution of the De

Delivery transactions were submitted to servers using the sa nism that other transactions used. The only difference wa asynchronous, i.e., control would return to the client proce deferred delivery part would complete asynchronously.

Clause 3 Related It

3.1 Transaction System Properties (ACID)

The results of the ACID tests must be disclosed along wi
requirements were met. This includes disclosing which case
Test 7.

The TPC Benchmark C Standard Specification defines a set of
ing system properties that a system under test (SUT) must
execution of the benchmark. Those properties are Atomicity
tion, and Durability (ACID).

This section defines each of those properties, describes the
they were present during the test and describes a series of
compliance with the specification.

3.2 Atomicity

The system under test must guarantee that transaction are
all individual operation on the data, or will assure that r
effects on the data.

3.2.1 Completed Transaction

Perform the Payment transaction for a randomly selected w
customer number as specified in Clause 2.5.1.2) and verify
DISTRICT, and WAREHOUSE tables have been changed appropria

The balances from a randomly selected warehouse, district,
retrieved by customer number. A Payment transaction was
completion of the Payment transaction, the balances of the
district, and customer were retrieved to verify that the ch

3.2.2 Aborted Transaction

addition, a History row representing the completion of the tr
from the database.

Perform the Payment transaction for a randomly selected wa
customer number as specified in Clause 2.5.1.2) and substitu
for the COMMIT of the transaction. Verify that the records i
and WAREHOUSE tables have NOT been changed.

The same warehouse, district, and customer used in the previ
execute a transaction that issued a ROLLBACK WORK command r
COMMIT WORK. After the transaction completed, the balances of
district, and customer rows were retrieved to verify that no
to the database. In addition, no new rows had been added to
to the aborted transaction.

3.3 Consistency

Consistency is the property of the application that require
to take the database from one consistent state to another,
in a consistent state.

The benchmark specification requires explicit demonstration
consistency conditions:

- the sum of the district balances in a warehouse is equal balance;
- for each district, the next order id minus one is equal id in the ORDER table and equal to the maximum new orde NEW-ORDER table;
- for each district, the maximum order id minus minimum or ORDER table plus one equals the number of rows in the NEW table for that district;
- for each district, the sum of the order line counts equals in the ORDER-LINE table for that district;

In order to demonstrate the consistency the following steps

- Prior to the start of a benchmark run, the consistency of verified by testing successfully conditions 1-4 describe
- A 21,500 user test with a checkpoint was executed.
- After completion of that test, the consistency of the dat by successfully testing the same consistency condition i

3.4 Isolation

Isolation can be defined in terms of phenomena that can o
transactions. These phenomena are P0 (Dirty Write), P1 (
Read), and P3 (Phantom). The table in Clause 3.4.1 of t
isolation requirements which must be met by the TPC-C tran
be enabled at either the system of application level to en

The requirements of the benchmark specification specify
isolation for New-Order, Payment, Delivery, and Order-Stat
Level 1 isolation for Stock-Level. ANSI SQL Level 3 isolati
Order, Payment, Delivery, and Order-Status in the TPC-C
following command:

- EXEC SQL SET ISOLATION TO REPEATABLE READ

The isolation level for Stock-Level was enabled via the fo

- EXEC SQL SET ISOLATION TO COMMITTED READ

The code for each of the above command may be found in App

The benchmark specification defines nine required tests
demonstrate that the required levels of transaction isolat
described in Clauses 3.4.2.1-3.4.2.9, were all performed an
Isolation Test 7, Case C was observed where the action on
to abort the incipient transaction and report an error to

3.5 Durability

The tested system must guarantee durability: the abilit
transactions and insure database consistency after recove
Clause 3.5.3.

List of single failures:

Permanent irrecoverable failure of any single durable medi
or recovery log data.

Instantaneous interruption (system crash/system hang) in p
to recover.

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

Three durability test were executed to satisfy the requirem
The test for loss of memory and instantaneous interruptio
performed with a fully scaled database with 21,500 emulated
and loss of data tests were performed on a ten warehouse data
users. To the best of our knowledge, these test prove

3.5.1 Loss of Data Disk

configuration used for the throughput test would also pass

The following steps were taken to demonstrate durability in t
disk:

1. The database was archived to tape.
2. The D_NEXT_O_ID fields for all rows in the district tabl
   up to determine the initial count of the total number orc
3. The benchmark was executed with 100 users. On the driv
   committed and rolled back New-Order transaction were re
   success file. After five minutes one of data disks was
4. The test was aborted on the driver and the data disk pc
5. The above data disk was overwritten with meaningless da
6. The data base was recovered using the tape archive crea
7. The contents of the success file on the driver and the
   compared to verify that records in the success file for
   Order transactions had corresponding records in the ORDER
   entries existed for rolled back transactions.
8. Step 2 was repeated to determine the total number of
   Count2 minus count1 equaled the number of committed Ne
   records in the success file.
9. Consistency Condition 3 of Clause 3.3.2.3 (no gaps in
   verified.

3.5.2 Loss of Log Disk

The following steps were taken to demonstrate durability in c
log disk:

1. The D_NEXT_O_ID fields for all rows in district table w
   to determine the initial count of the total number of orc
2. The benchmark was executed with 100 users. On the driv
   committed and rolled back New-Order transaction were re
   success file.
3. At full load, the recovery log disk was powered off.
4. Because of the Informix mirroring feature enabled for th
   system continued to process transactions.
5. The test completed normally.
6. The contents of the success file on the driver and the
   compared to verify that records in the success file for

Order transactions had corresponding records in the ORDER entries existed for rolled back transactions.

7. Step 1 was repeated to determine the total number of Count2 minus count1 equaled the number of committed N records in the success file.

8. Consistency Condition 3 of Clause 3.3.2.3 (no gaps ir verified.

3.5.3 Instantaneous Interruption and Loss of Memory

Instantaneous interruption and loss of memory tests were c loss of power erases the contents of memory. This failure wa the primary power to the System Under Test while the benchr

1. The D_NEXT_O_ID fields for all rows in district table to determine the initial count of the total number of or

2. The benchmark was executed with Full User Count users system, the committed and rolled back New-Order trans recorded in a success file. After five minutes, one of removed.

3. At full load the system s primary power was turned off

4. The test was aborted on the driver system.

5. Power was restored to the system, the system rebooted recovery performed. The database recovery restores the da just after the last committed transaction before the fai

6. The contents of the success file on the driver and th compared to verify that records in the success file fo Order transactions had corresponding records in the ORDER entries existed for rolled back transactions.

7. Step 1 was repeated to determine the total number of Count2 minus count1 was not less than the number of comm Order records in the success file.

8. Consistency Condition 3 of Clause 3.3.2.3 (no gaps ir verified.

Clause 4 Related It

4.1 Initial Cardinality of Tables

The Cardinality (e.g. number of rows) of each table, as it
(see Clause 4.2), must be disclosed. If the database was
WAREHOUSE table were delete (see Clause 4.2.2) the cardinali
as initially configured and the number of rows deleted mus

The TPC-C database for this test was initially configured wit
to the measurement, 50 rows were deleted from the Warehouse t
of each table in the database is listed in Table 4.1.

Table 4.1: Initial Cardinality of Database Tables

Table	Occurrences
Warehouse	2,200
District	22,000
Customer	66,000,000
History	66,000,000
Order	66,000,000
New Order	19,800,000
Order Line	659,918,692
Stock	220,000,000
Item	100,000

4.2 Database Layout

The distribution of tables and logs across all media must be
systems.

Tables 4.2 and 4.3 list the distribution of the database over
and priced configurations. The benchmark configuration contai
GB each. The priced configuration contained 576 disks, of the
the 180-day storage requirement.

4.3 DBMS: Data Model and DBMS Interface/ Access Language

A statement must be provided that describes:

1. The data model implemented by the DBMS used (e.g., rela

2. The database interface (e.g., embedded, call level) and ac
read/write) used to implement the TPC-C transactions. If mor
is used to implement TPC-C, each interface/access language
interface/access language is used with which transaction ty

INFORMIX-OnLine Dynamic Server is a relational DBMS. A com
dynamic SQL and stored procedures were used.

4.4 DBMS Partitions/ Replications

The mapping of database partitions/replications must be exp

No table partitioning or replication was done.

4.5 DBMS Space Requirements

Details of the 180 day space computation along with proof th
8 hours of growth for dynamic tables (Order, Order-Line, and
4.2.3).

Appendix D lists the space requirements for the 180-day spac
log space for eight hours.

Table 4.2: Benchmark Configuration Disk Storage Detail

Table 4.2: Benchmark Configuration Disk Storage Detail (cont.)

Table 4.2: Benchmark Configuration Disk Storage Detail (cont.)

Table 4.3: Priced Configuration Disk Storage Detail

Table 4.3: Priced Configuration Disk Storage Detail (cont.)

Table 4.3: Priced Configuration Disk Storage Detail (cont.)

Table 4.3: Priced Configuration Disk Storage Detail (cont.)

Table 4.3: Priced Configuration Disk Storage Detail (cont.)

Clause 5 Related It

5.1 Measured Throughput (tpmC)

Measured tpmC must be reported.

The measured tpmC was 25,309.20.

5.2 Response Times

Ninetieth percentile, maximum and average response times types as well as for the Menu response time.

Table 5.1: Response Time Data

Transaction	Average	90th Percentile	Maximum
New-Order	1.62	1.70	52.19
Payment	1.38	1.94	62.91
Order-Status	0.91	0.95	41.10
Delivery (interactive)	0.32	0.42	0.74
Delivery (deferred)	16.38	47.01	144.00
Stock-Level	1.57	3.06	44.55
Menu:			
New Order	0.34	0.41	20.09
Payment	0.34	0.41	5.84
Order Status	0.34	0.41	2.95
Delivery	0.34	0.41	0.72
Stock Level	0.34	0.41	2.28

5.3 Keying and Think Times

The minimum, the average, and the maximum keying and think transaction type.

Table 5.2: Keying Times

Transaction Type	Minimum	Average	Maximum
New-Order	18.00	18.00	18.00
Payment	3.00	3.00	3.00
Order-Status	2.00	2.00	2.00
Delivery (interactive)	2.00	2.00	2.00
Stock-Level	2.00	2.00	2.00

Table 5.3: Think Times

Transaction Type	Minimum	Average	Maximum
New-Order	0.10	12.05	120.00
Payment	0.10	12.03	120.00
Order-Status	0.10	10.06	100.00
Delivery (interactive)	0.10	5.06	50.00
Stock-Level	0.10	5.05	50.00

5.4 Response Time Frequency Distribution Curves

Response Time frequency distribution curves (see Clause transaction type.

5.5 Response Time vs. Throughput Curve

The performance curve for response times versus throughput for the New-Order transaction.

5.6 Think Time Frequency Distribution Curves

A Think Time frequency distribution curve (see Clause 5.6.3 transaction.

5.7 New-Order Throughput vs. Time

A graph of throughput versus elapsed time (see Clause 5.6.5 transaction.

The above graphs are presented below in Figures 5.1 - 5.8.

Figure 5.1: New Order Response Time Distribution

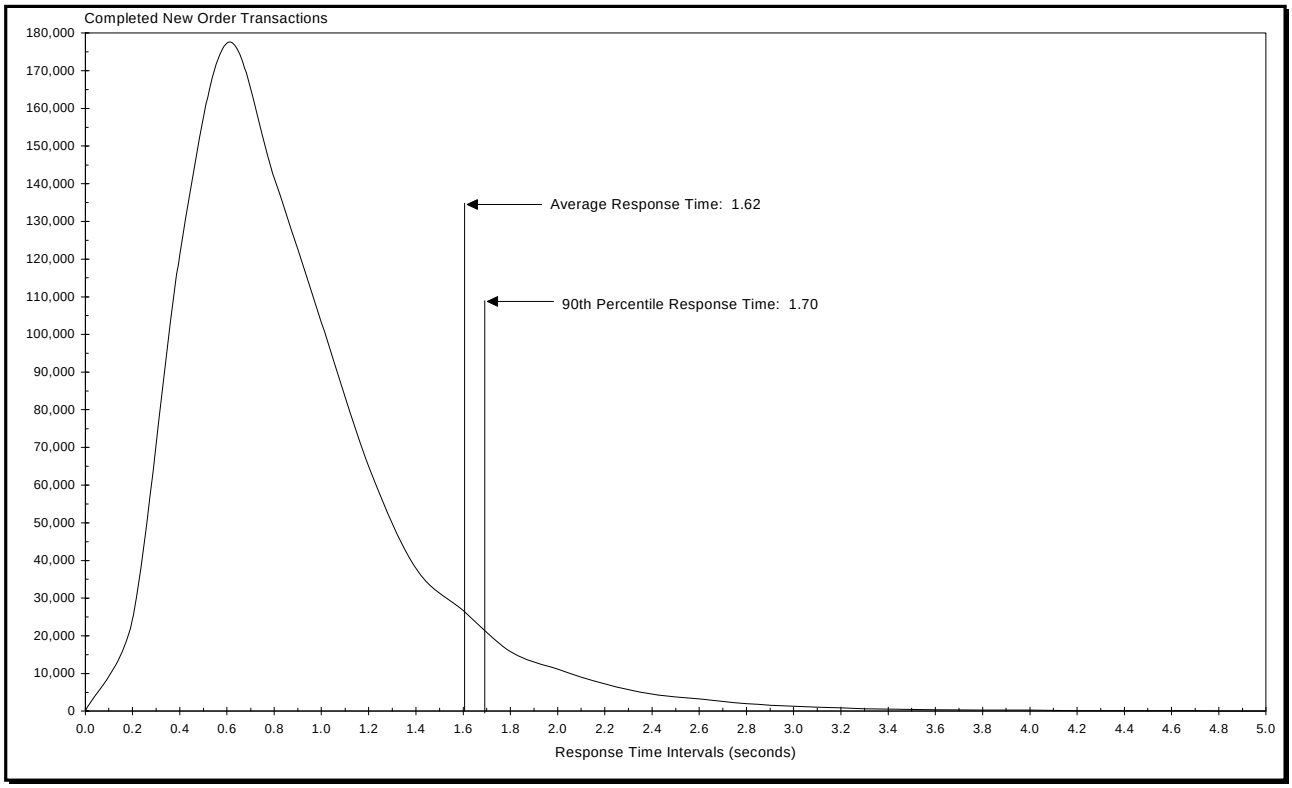


Figure 5.2: Payment Response Time Distribution

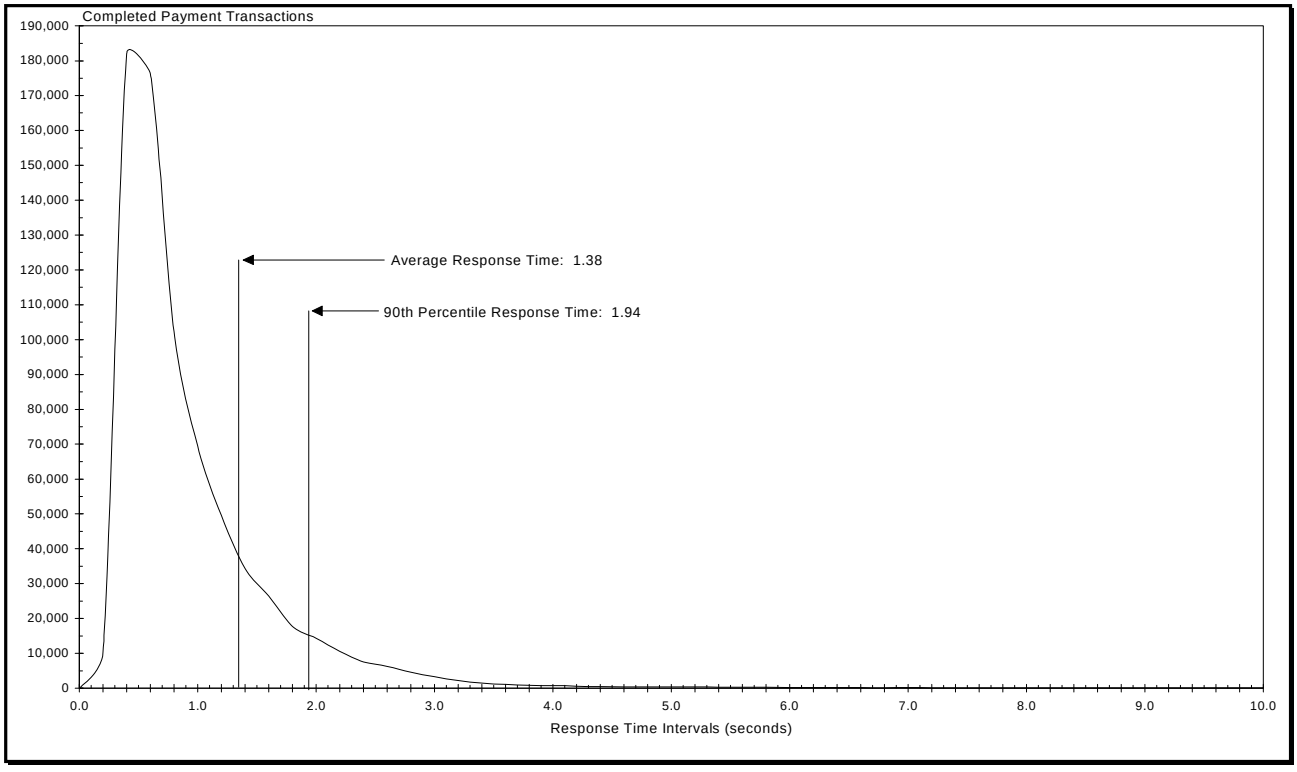


Figure 5.3: Order-Status Response Time Distribution

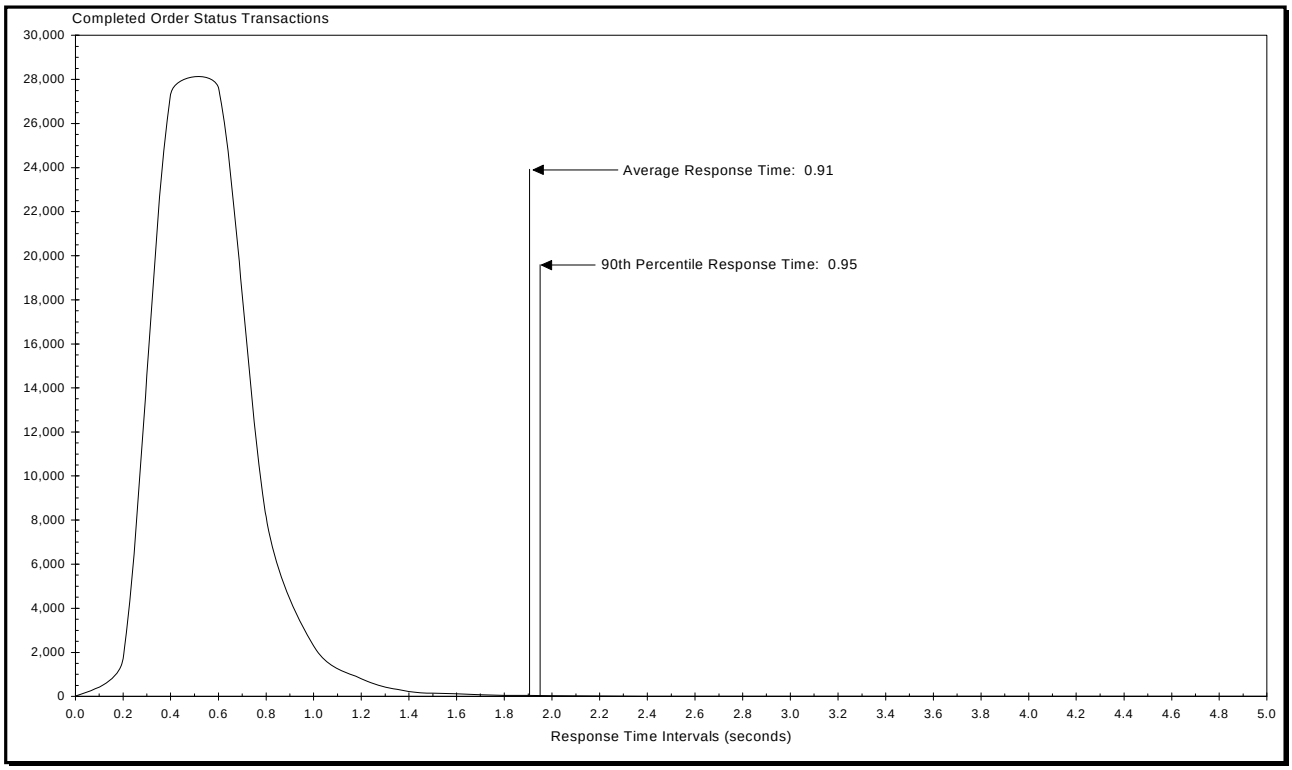


Figure 5.4: Delivery Response Time Distribution

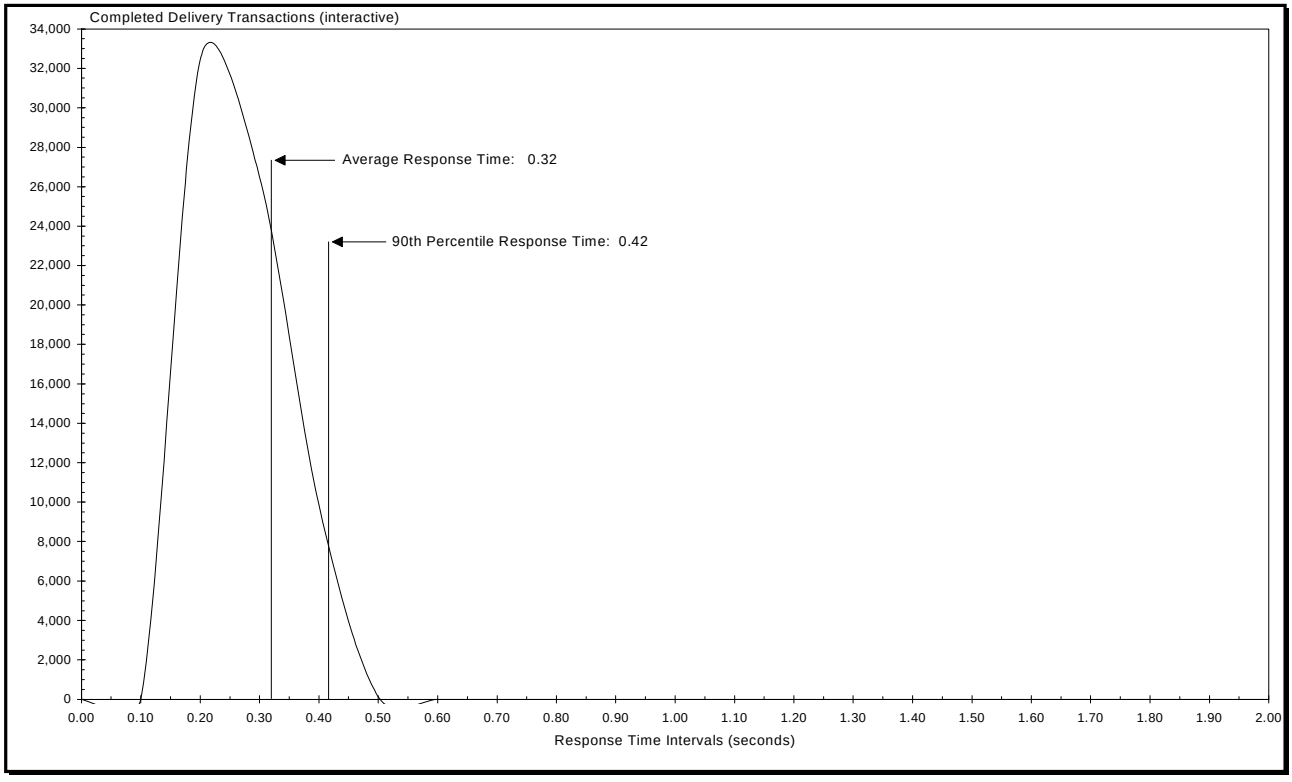


Figure 5.5: Stock-Level Response Time Distribution

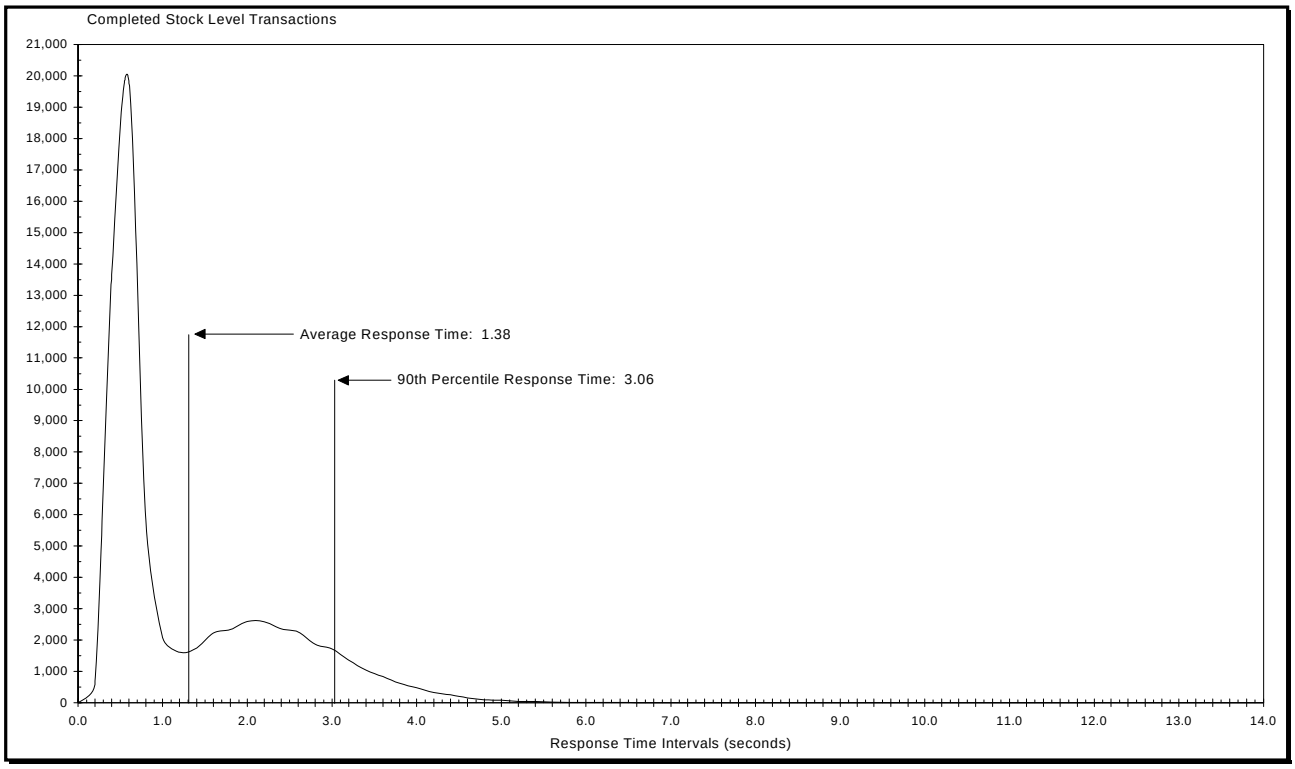


Figure 5.6: Response Time versus Throughput (tpmC)

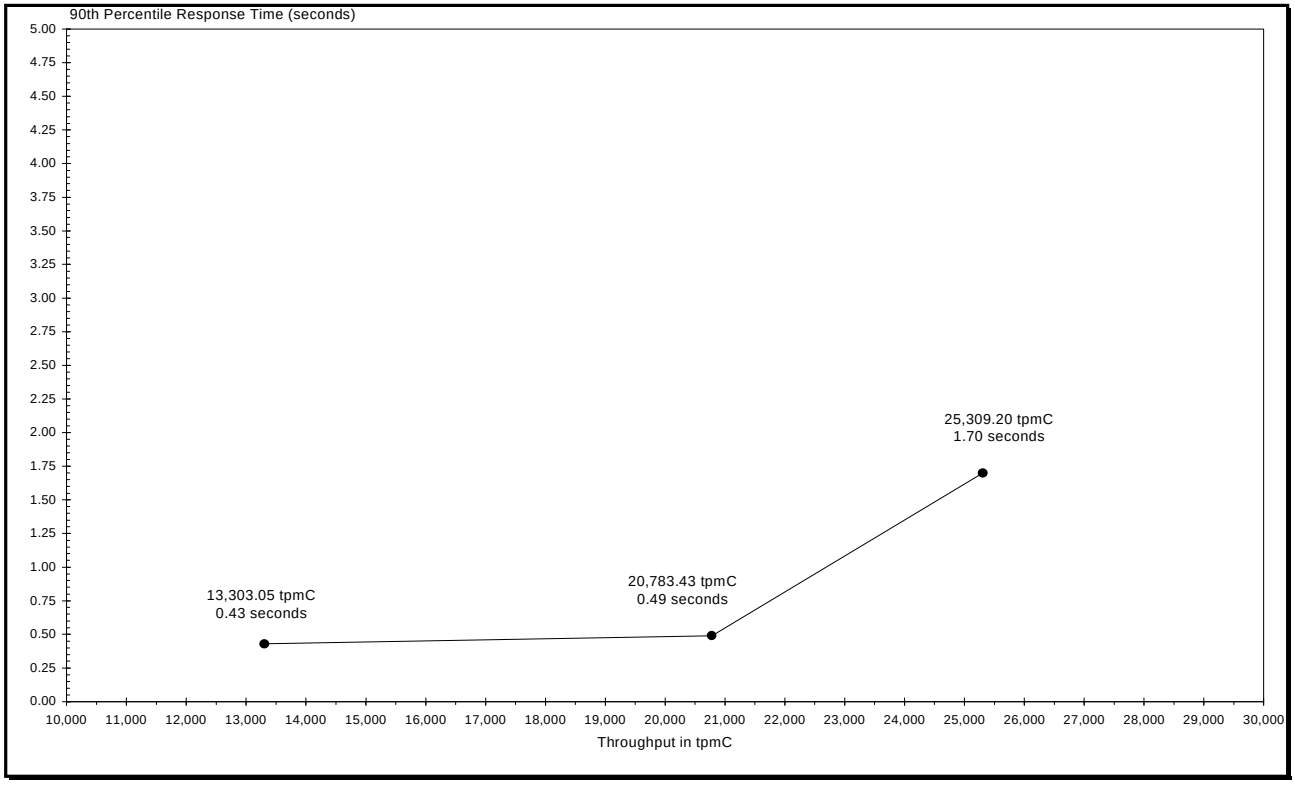


Figure 5.7: New-Order Think Time Distribution

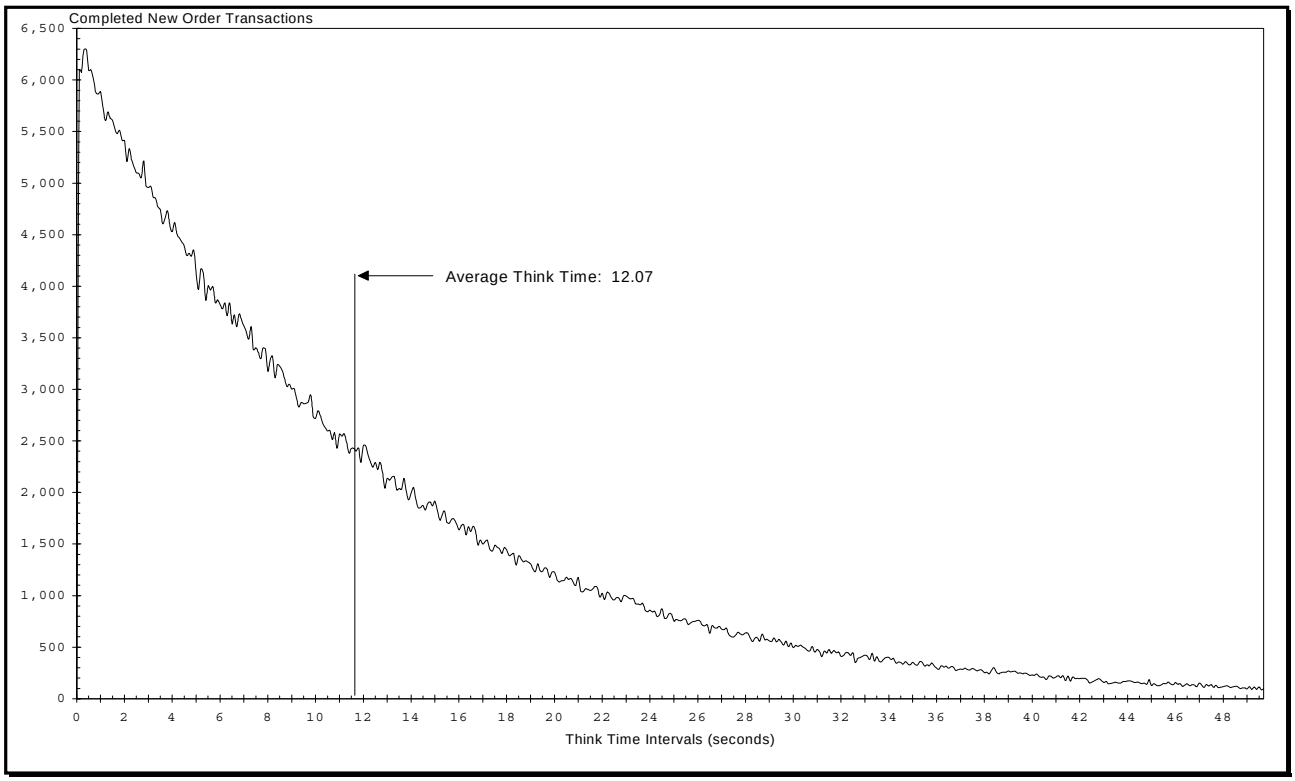
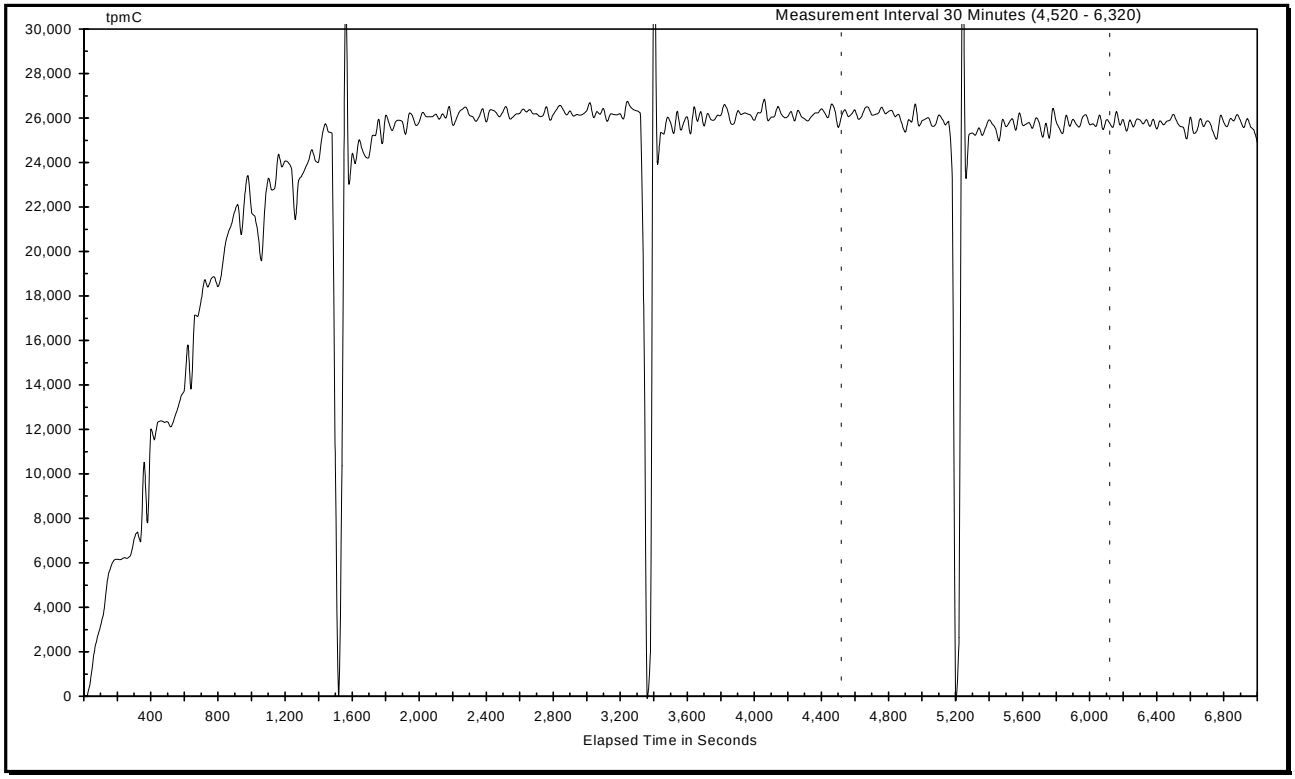


Figure 5.8: Throughput (tpmC) versus Time



5.8 Determination of 'Steady State'

The method used to determine that the SUT had reached a st
measurement interval (see Clause 5.5) must be described.

The transaction throughput rate (tpmC) and response time we
after the initial ramp up period. The throughput and resp
determined by examining data reported for each 30-second
duration of the benchmark. Ramp-up, steady state, and ram
discernible in the graph presented in Figure 5.8.

5.9 Work Performed During 'Steady State'

A description of how the work normally performed during
checkpointing, writing redo/undo log records, etc.), actu
interval must be reported.

Table updates are buffered, and a collection of page-clean
nously writes dirty pages to disk.

An INFORMIX OnLine checkpoint writes all buffers in memory
data on disk matches what is in memory. Checkpoints are mark
written into the logs.

INFORMIX-OnLine executes a checkpoint for the following co

1. If database activity has occurred without a checkpoint,
   the tbconfig file, specifies the interval in seconds bet

2. When the physical log becomes seventy-five percent fu
   automatically executed. The physical log is emptied.

3. Issuing the tbmode-c command explicitly forces a checkp

During each benchmark measurement, a background process, p
issued an initial checkpoint (tbmode -c) immediately prior
and then a second after all users are active. After the
subsequent checkpoints were a result of the ckptintvl par
INFORMIX OnLine tbconfig file: 1,800 seconds. The result
initial checkpoint, a checkpoint was executed every thirty
log size was chosen such that it would not become 75% full
checkpoint interval had ended.

5.10 Reproducibility

A description of the method used to determine the reprodu

In a repeat test, a throughput of 25,211.50 tpmC was achieve
during a 30-minute, steady state run.

5.11 Measurement Interval Duration

A statement of the duration of the measurement interval fc
Throughput (tpmC) must be included.

The measurement interval was 30 minutes.

5.12 Regulation of Transaction Mix

The method of regulation of the transaction mix (e.g. card
must be described. If weighted distribution is used and t
with each transaction type, the maximum adjustments to the
be disclosed.

The weighted random distribution method of Clause 5.2.4.
distribution was not adjusted during the run.

5.13 Transaction Mix

The percentage of the total mix for each transaction type

Table 5.4: Transaction Mix

New-Order	44.66%
Payment	43.17%
Order-Status	4.06%
Delivery	4.05%
Stock-Level	4.06%

5.14 Transaction Statistics

The percentage of New-Order transactions rolled back as a
be disclosed. The average number of order-lines entered
disclosed. The percentage of remote order-lines entered
disclosed. The percentage of remote Payment transactions
customer selections by customer last name in the Payment ar
disclosed. The percentage of Delivery transactions skippec
orders in the New-Order table must be disclosed.

See Table 2.1 (8.1.6.15-20, quoted above, specify the same
10).

5.15 Checkpoint Statistics

The number of checkpoints in the measurement interval, the
measurement interval to the first checkpoint, and the Chec

There is one checkpoint in the measurement interval, beginni
seconds from the start of the measurement window. The che
1800 seconds. The measurement interval is 1800 seconds. I
Clause 5.5.2.2. there is no checkpoint within a span of 180
after the beginning or end of the measurement interval.

Clause 6 Related It

6.1 Remote Terminal Emulator (RTE) Description

If the RTE is commercially available, then its inputs must
must be supplied of what inputs (e.g., scripts) to the RTE

Appendix E contains the portions of the RTE scripts that sel
generate the transaction input data.

6.2 Emulated Components

It must be demonstrated that the functionality and perform
in the Driver System are equivalent to the priced system.

There were no emulated components in the benchmark configura
emulated users workstations.

6.3 Functional Diagrams

A complete functional diagram of both benchmark and the co
system must be disclosed. A detailed list of all hardware
on the Driver System and its interface to the SUT must be

Figures 0.1 and 0.2 (in Section 0) show functional diagrams
configured systems. A description of the RTE and benchmark
above.

6.4 Network

The network configuration of both the tested and proposed s
a thorough explanation of exactly which parts are being rep
disclosed.

Figures 0.1 and 0.2 (in Section 0) also diagram the network
benchmark and configured systems, and represent the Driver
replacing the user workstations that are directly connected

The bandwidth of the networks used in the tested/priced co

Ethernet local area networks (LAN) with a bandwidth of 10 meg
used in the tested/priced configurations.

Clause 7 Related It

7.1 Pricing

A detailed list of hardware and software used in the priced
have vendor part number, description, and release/revision
or committed delivery data. If package-pricing is used cont
source(s) and effective date(s) of price(s) must also be r

The total 5-year price of the entire configuration must be
and maintenance charges. Separate component pricing is re
used must be disclosed.

7.1.1 System Pricing

The priced configuration consists of an integrated system pac
and components. Prices for all Silicon Graphics products a
by 1DIRECT as listed in Appendix F.

7.1.2 Support Pricing

Five years of Informix support is priced for both INFOR
INFORMIX-ESQL/C. Where appropriate, five years of maintenan
Silicon Graphics products.

7.3 Availability

The committed delivery date for general availability (avai
calculation must be reported. When the priced system incl
dates, the reported availability date for the priced syste
are committed to be available.

The hardware, software, and support/maintenance products pric
are detailed on page vi.

All products used in this benchmark will be available Octob

7.4 Measured tpmC and Price/Performance

A statement of the measured tpmC as well as the respective ca
performance (price/tpmC), and the availability date must be

The maximum Qualified Throughput for the Silicon Graphics Orig
INFORMIX-OnLine Dynamic Server, version 7.3.UC1 was 25,309
$139.04 per tpmC. All products used in this benchmark will be
1997.

Clause 8 Related It

Clause 8 specifies the composition of this full disclosure

Clause 9 Related It

Auditor's Report

The auditor s name, address, phone number, and a copy of tl
compliance must be included in the Full Disclosure Report

This implementation of the TPC Benchmark C on the Silicc
gin2000 Server using INFORMIX-OnLine Dynamic Server 7.3.UC
by Richard Gimarc, a TPC certified auditor, of:

Performance Metrics, Inc.
2229 Benita Drive
Suite 101
Rancho Cordova, CA 95670
Telephone: (916) 635-2822
Fax: (916) 858-0109

The auditor s attestation letter is shown on the next pag

PERFORMANCE METRICS INC.
TPC Certified Auditors

April 28, 1997

Cuong Tran
Manager, Database Performance Group
Silicon Graphics Computer Systems
2011 N. Shoreline Blvd.
Mountain View, CA 94039

Walter E. Baker
Director, Performance Engineering
Informix Software, Inc.
4100 Bohannon Drive
Menlo Park, CA 94025

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

Platform:	Silicon Graphics Origin2000 Server
Database Manager:	INFORMIX-OnLine Dynamic Server 7.3
Operating System:	IRIX 6.4 S2MP
Transaction Manager:	BEA Tuxedo 6.1 CFS

CPUs	Memory	Disks	New-Order Response Time @ 90%	tpmC
Server: Silicon Graphics Origin2000 Server				
30 MIPS R10000 @ 195 MHz	Main: 13 GB L1 Cache: 32 KB2 L2 Cache: 4 MB	421 @ 4.2 GB	1.70 sec.	25,309.20
26 Clients: Silicon Graphics Challenge S Server				
1 MIPS R4400 @ 200 MHz	Main: 256 MB L1 Cache: 16 KB L2 Cache: 1 MB	1 @ 0.99 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, including error handling.
- The database files were properly sized and populated.
- The database was properly scaled with 2,200 warehouses. Only 2,150 warehouses were active during measurement.

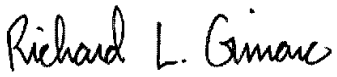
2229 Benita Drive, Suite 101, Rancho Cordova, CA 95670
(916) 635-2822 Fax: (916) 858-0109 e-mail: Lorna@PerfMetrics.com

PERFORMANCE METRICS INC.
TPC Certified Auditors

- The ACID properties were met, including phantom protection.
- The durability data loss and log loss tests were performed on a 10-warehouse database.
- Input data was generated according to the specified percentages.
- Eight hours of mirrored log space was configured on the measured system.
- The data for the 180-day space calculation was verified.
- The priced server configuration contains 28 CPUs. It was demonstrated that two of the 30 CPUs seen in the server hardware configuration were restricted from use and not active during measurement.
- Measurement cycle times did not include any 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 21,500 user contexts present on the system.
- Each emulated user had a different random number stream.
- The NURand constants used for database load and at run time were verified.
- System pricing was checked for major components and maintenance.

Additional Audit Notes:

Regards,



Richard L. Gimarc
Auditor

2229 Benita Drive, Suite 101, Rancho Cordova, CA 95670
(916) 635-2822 Fax: (916) 858-0109 e-mail: Lorna@PerfMetrics.com

Report Availabi

Requests for the TPC Benchmark C Full Disclosure Report for Origin2000 Server using INFORMIX-OnLine Dynamic Server Ver should be sent to:

TPC
777 N. First Street
Suite 600
San Jose, CA 95112 6311

or

Informix Software, Inc.
Performance Engineering
Attention: Cynthia L. Beaudett
4100 Bohannon Drive
Menlo Park, CA 94025

Appendix A - Client/Server

databuf.h

```
#ifndef __TPCC_DATABUF_H__
#define __TPCC_DATABUF_H__

/*
 * BEWARE
 *
 * it's no use using #define for the lenght of the arrays,
 * because that file is also include thru 'SQL INCLUDE' and
 * therefore not pre-processed.
 */

struct data_header {
        short int       dtype;
        short int       returncode;
        int                             sql_code;
        int                             isam_code;
};

struct OL_TABLE {
        short int       ol_supply_w_id;
        short int       ol_quantity;
        short int       s_quantity;
        int                             ol_i_id;
        char                    name_i[25];
        char                    brand_generic;
        double                  price;
        double                  ol_amount;
};

struct OL_TABLE2 {
        double                  ol_amount;
        int                             ol_i_id;
        short int       ol_supply_w_id;
        short int       ol_quantity;
        char                    delivery_date[20];
};

struct NEWO_DATA {
        struct data_header header;
        short int       w_id;
```

```
        short int       d_id;
        int                             c_id;
        short int       o_ol_cnt;
        short int       o_all_local;
        short int       items_valid; /* true if all valid */
        int                             o_id;
        double                  w_tax;
        double                  d_tax;
        double                  total;
        double                  c_discount;
        char                    new_date[20];
        char                    c_last[17];
        char                    c_credit[3];
        struct OL_TABLE ol_table[15];
};

struct PMT_DATA {
        struct data_header header;
        short int       w_id;
        short int       d_id;
        double                  h_amount;
        double                  c_credit_lim;
        double                  c_balance;
        double                  c_discount;
        double                  c_ytd_payment;
        int                             c_id;
        short int       c_w_id;
        short int       c_d_id;
        short int       byname;
        char                    pay_date[20];

        char                    w_name[11];
        char                    w_street_1[21];
        char                    w_street_2[21];
        char                    w_city[21];
        char                    w_state[3];
        char                    w_zip[10];

        char                    d_name[11];
        char                    d_street_1[21];
        char                    d_street_2[21];
        char                    d_city[21];
        char                    d_state[3];
        char                    d_zip[10];

        char                    c_first[17]; /* was
C_LAST_LEN already includes +1 */
        char                    c_middle[3];
        char                    c_last[17];
        char                    c_phone[17];
        char                    c_credit[3];
        char                    c_street_1[21];
        char                    c_street_2[21];
        char                    c_city[21];
        char                    c_state[3];
        char                    c_zip[10];
        short int       c_payment_cnt;
        char                    c_since[20];
        char                    c_data[200];
};

struct ORDS_DATA {
        struct data_header header;
        short int       w_id;
        short int       d_id;
        int                             c_id;
        int                             o_id;
        short int       o_ol_cnt;
        short int       byname;
        short int       o_carrier_id;
        char                    c_last[17];
        char                    c_first[17];
        char                    c_middle[3];
        char                    cur_date[20];
        double                  c_balance;
        struct OL_TABLE2 ol_table[15];
};

struct DVRY_DATA {
        struct data_header header;
        short int       w_id;
        short int       carrier_id;
        int                                     startq_sec;
        int                                     startq_usec;
};

struct STKL_DATA {
        struct data_header header;
        short int       w_id;
        short int       stkl_d_id;
        short int       threshold;
        int                                     stock_count;
};

#endif
```

utils.h

```
#ifndef __TPCC_UTILS__
#define __TPCC_UTILS__

#define YEAR_TO_DATE   1
#define YEAR_TO_SECOND 2
#define MAX_STR_LEN    128

#define ERROR(x) {fprintf(stderr,"Error: %s\n",x);exit(-1);}

extern void convert_datetime();
extern double get_time();

#ifdef NULL_TRANS
void init_ms_sleep();
void ms_sleep();
#endif

#endif
```

tuxclient.h

```
/*****************************************************************/
/*
        */
/*      File: tuxclient.h
*/
/*
        */
/*****************************************************************/

/*-------------------------------------------------------------
------*/
/*      include files
        */
/*-------------------------------------------------------------
------*/
#include “atmi.h”              /* TUXEDO atmi library */

/*-------------------------------------------------------------
------*/
/*      global variables
*/
/*-------------------------------------------------------------
------*/
int             transtatus;
char    *TMbuffer;
long    TMinbufsize, TMoutbufsize;

#define TMINBUFSIZE     1536

/*-------------------------------------------------------------
------*/
/*      connect to TM                           */
/*-------------------------------------------------------------
------*/
connect_to_TM ()
{
        TMinbufsize = TMINBUFSIZE;

        if (tpinit((TPINIT *)NULL) == -1)
        {
                printf(“rtn: tpinit(), tperrno: %d \n”, tperrno);
                exit(1);
        }

        if ((TMbuffer = tpalloc(“CARRAY”, NULL, TMinbufsize))
== NULL)
        {
                printf(“rtn: tpalloc(), tperrno: %d \n”, tperrno);
                exit(2);
        }
}

/*-------------------------------------------------------------
------*/
/*      disconnect from TM                              */
/*-------------------------------------------------------------
------*/
disconnect_from_TM ()
{
        if (tpterm() == -1)
        {
                printf(“rtn: tpterm(), tperrno: %d \n”, tperrno);
                exit(3);
        }
}

/*-------------------------------------------------------------
------*/
/*      submit TM tran
        */
/*-------------------------------------------------------------
------*/
submit_TM_tran (servname,size)
char    *servname;
int     size;
{
        transtatus = tpcall(servname, TMbuffer, size, &TMbuffer,
                &TMoutbufsize, 0);
        if (transtatus == -1)
        {
                char    str[60];

                sprintf(str, “rtn: tpcall(), tperrno: %d,
transtatus: %d \n”,
                        tperrno, transtatus);
                fprintf(stderr, str);
                return(-1);
        }
        return(0);
}

/*-------------------------------------------------------------
------*/
/*      submit TM asynchronous transaction
*/
/*-------------------------------------------------------------
------*/
submit_TM_atran (servname,size)
char    *servname;
int     size;
{
        transtatus = tpacall(servname,TMbuffer,size,
TPNOREPLY);
        if (transtatus == -1)
        {
                char    str[60];

                sprintf(str, “rtn: tpacall(), tperrno: %d,
transtatus: %d \n”,
                        tperrno, transtatus);
                fprintf(stderr, str);
                return(-1);
        }
        return(0);
}
```

ex-trans.h

```
#ifndef __EX_TRANS_H__
#define __EX_TRANS_H__

#define I_NAME_LEN      24
#define I_DATA          50
#define W_NAME_LEN      10
#define ADDR_LEN        20
#define STATE_LEN       2
#define ZIP_LEN         9
```

```
#define DIST_INFO_LEN      24
#define S_DATA_LEN         50
#define D_NAME_LEN         10
#define H_DATA_LEN         24
#define CARRIER_LEN        2

#define C_LAST_LEN         17
#define C_MID_LEN          2
#define PHONE_LEN          16
#define CREDIT_LEN         2

#define C_DATA_LEN  500
#define BC_DATA_LEN  23

#define DATETIME_LEN       19

#define MAX_OL             15
#define C_LAST_LEN         17

#endif
```

do_tpcc.h

```
#ifndef __DO_TPCC_H__
#define __DO_TPCC_H__

#define MAX_OL                 15     /* Maximum items in an order
*/
#define MIN_OL                 5
#define MAX_FLDS       200     /* Maximum fields in a TPC-C form
*/
#define D_PER_W        10

#define ON             1
#define OFF            0

#define TRUE           1
#define FALSE          0

#define YES            1
#define NO             0

#define INSIZE  1024

#define RETRY_MSG      "Try again..."
#define COMMIT_NEWO                    99
#define COMMIT_PAYMENT         88
#define COMMIT_DELIVERY        77
#define COMMIT_ORDERSTAT       66
#define COMMIT_STOCKLEV                55

#define INVALID_NEWO  1
#define SQL_ERROR     -1
#define OS_ERROR      -2
#define CANCEL        -1

#endif
```

infdatab.h

```
#ifndef __TPCC_DATABUF_H__
#define __TPCC_DATABUF_H__
/*
#include "utils.h"
#include "ex_trans.h"
*/
#define MAX_OL             15
#define C_LAST_LEN         17

#define I_NAME_LEN         24
#define I_DATA             50
#define W_NAME_LEN         10
#define ADDR_LEN           20
#define STATE_LEN          2
#define ZIP_LEN            9
#define DIST_INFO_LEN      24
#define S_DATA_LEN         50
#define D_NAME_LEN         10
#define H_DATA_LEN         24
#define CARRIER_LEN        2

#define C_LAST_LEN         17
#define C_MID_LEN          2
#define PHONE_LEN          16
#define CREDIT_LEN         3

#define C_DATA_LEN  500
#define BC_DATA_LEN  23

#define DATETIME_LEN       20

#define THEIR_NEW_ORDER        1
#define THEIR_PAYMENT          2
#define THEIR_ORDER_STATUS     3
#define THEIR_DELIVERY         4
#define THEIR_STOCK_LEVEL      5


#endif

struct data_header {
    short int  dtype;
    short int  returncode;
    int        sql_code;
    int        isam_code;
};

struct OL_TABLE {
        short int  ol_supply_w_id;
        short int  ol_quantity;
        short int  s_quantity;
             int  ol_i_id;
        char       name_i[I_NAME_LEN + 1];
        char       brand_generic;
        double     price;
        double     ol_amount;
};

struct OL_TABLE2 {
        double     ol_amount;
        int       ol_i_id;
        short int  ol_supply_w_id;
```

```
        short int  ol_quantity;
        char       delivery_date[20];
};

struct NEWO_DATA {
                struct data_header header;
                short int       w_id;
                short int       d_id;
                     int       c_id;
                short int       o_ol_cnt;
                short int       o_all_local;
                short int       items_valid; /* true if all valid */
                int             o_id;
        double          w_tax;
                double          d_tax;
                double          total;
                double          c_discount;
/* sync with tpccfs - Gerson 08/30/96
                char            new_date[DATETIME_LEN + 1];
*/
                char            new_date[DATETIME_LEN];
                char            c_last[C_LAST_LEN];
/* sync with tpccfs - Gerson 08/30/96
                char            c_credit[CREDIT_LEN + 1];
*/
                char            c_credit[CREDIT_LEN ];
                struct OL_TABLE ol_table[MAX_OL];
};

struct PMT_DATA {
                struct data_header header;
        short int w_id;
        short int d_id;
        double    h_amount;
        double    c_credit_lim;
        double    c_balance;
        double    c_discount;
        double    c_ytd_payment;
             int c_id;
        short int c_w_id;
        short int c_d_id;
        short int byname;
        char      pay_date[20];

        char      w_name[11];
        char      w_street_1[21];
        char      w_street_2[21];
        char      w_city[21];
        char      w_state[3];
        char      w_zip[10];

        char      d_name[11];
        char      d_street_1[21];
        char      d_street_2[21];
        char      d_city[21];
        char      d_state[3];
        char      d_zip[10];

        char      c_first[17]; /* was C_LAST_LEN already includes
+1 */
        char      c_middle[3];
        char      c_last[17];
        char      c_phone[17];
        char      c_credit[3];
        char      c_street_1[21];
        char      c_street_2[21];
        char      c_city[21];
        char      c_state[3];
        char      c_zip[10];
        short int c_payment_cnt;
        char      c_date[20];
        char      c_data[200];
};

struct ORDS_DATA {
                struct data_header header;
        short int w_id;
        short int d_id;
             int c_id;
             int o_id;
        short int o_ol_cnt;
        short int byname;
        short int o_carrier_id;
        char      c_last[17];
        char      c_first[17];
        char      c_middle[3];
        char      cur_date[20];
        double    c_balance;
        struct OL_TABLE2 ol_table[15];
};

struct DVRY_DATA {
                struct data_header header;
                short int w_id;
                short int carrier_id;
        int     startq_sec;   /* new */
        int     startq_usec;   /* new */
/*              long  cltinit;            old */
/*              double start_queue;       old */
};

struct STKL_DATA {
                struct data_header header;
                short int w_id;
                short int stkl_d_id;
                short int threshold;
        int       stock_count;
};
```

clientlog.c

```
/
*******************************************************************************
****************************** clientlog.c
************************************
*******************************************************************************/
/*
**  clientlog.c — Routine for writing out messages form client
processes -
**                          useful for detailed error reporting and for
debugging
*/

#include <stdio.h>
#include <stdarg.h>

#define BACKTAB          2     /** Decided to use the CTRL B for
now **/
#define DELETE          127
#define ESCAPE           27
#define LF              10
```

```
#define QUIT            3       /** CNTRL-C Key stroke to quit for
now **/
#define SPACE           32
#define SUBMIT          13      /** Done with screen and submit
key: CR  **/
#define TAB             9
#define RTE_SYNCH_CHARACTER '\1'
/*
#define TPCC_HOME "/usr/users/sybase/kits/tpcc"
*/

    static FILE *clientlog;
    static int Clog_open=0;

void Clog(char *fmt, ...)

{
    char tmpfname[256];
#ifdef SCO
    char     fname[]="/usr/sybase/kits/tpcc/CLIENTLOG";
#else
    char    fname[]="CLIENTLOG";
#endif
    va_list argp;

    if (!Clog_open) {
/*
        sprintf(fname,"%s/
%s.%d",getenv("TPCC_HOME"),"CLIENTLOG",getpid());
        sprintf(fname,"%s/
CLIENTLOG.%d",TPCC_HOME,getpid());
*/

        sprintf(tmpfname,"%s/
CLIENTLOG.%d",getenv("TPCC"),getpid());
        clientlog = fopen(tmpfname,"w");
                        if ( clientlog ==  NULL ) {
                                printf ( " Can't open %s\n",
tmpfname);
                                exit(0);
                        }
        Clog_open = 1;
    }


    va_start(argp, fmt);
    vfprintf(clientlog, fmt, argp);
    va_end(argp);
    fflush(clientlog);
}


void SCREENlog (int *flag,char *screen)
{
    char fname[256];
    int  i,char_ct;

    if (!Clog_open) {
        sprintf(fname,"%s/
%s.%d",getenv("TPCC_HOME"),"CLIENTLOG",getpid());
        clientlog = fopen(fname,"w");
        Clog_open = 1;
    }
    fprintf(clientlog,"** %d **\n",flag);
    char_ct = 0;
    fprintf(clientlog,"SCR: ");
    for (i = 0; screen[i] != 0; char_ct++,i++) {
        switch (screen[i]) {
            case BACKTAB:   fprintf(clientlog,"<BACKTAB>");
break;
            case DELETE:    fprintf(clientlog,"<DEL>"); break;
            case ESCAPE:    fprintf(clientlog,"<ESC>"); break;
            case LF:        fprintf(clientlog,"<LF>"); break;
            case QUIT:      fprintf(clientlog,"<^C>");break;
            case SUBMIT:    fprintf(clientlog,"<CR>"); break;
            case TAB:       fprintf(clientlog,"<TAB>"); break;
            case RTE_SYNCH_CHARACTER:
                        fprintf(clientlog,"<^A>"); break;
            default:        fprintf(clientlog,"%c",screen[i]);
        }
        if (char_ct > 192) {
            char_ct = 0;
/*            fprintf(screenlog,"\n"); */
        }
    }
    fprintf(clientlog,"\n");
    fflush(clientlog);
}

void SCREENlog2 (char *screen,int len)
{
    char fname[256];
    int  char_ct;

    if (!Clog_open) {
        sprintf(fname,"%s/
%s.%d",getenv("TPCC_HOME"),"CLIENTLOG",getpid());
        clientlog = fopen(fname,"w");
        Clog_open = 1;
    }
    fprintf(clientlog,"SCR: %d chars: ",len);
    for (char_ct = 0; char_ct < len; char_ct++) {
        switch (screen[char_ct]) {
            case BACKTAB:   fprintf(clientlog,"<BACKTAB>");
break;
            case DELETE:    fprintf(clientlog,"<DEL>"); break;
            case ESCAPE:    fprintf(clientlog,"<ESC>"); break;
            case LF:        fprintf(clientlog,"<LF>"); break;
            case QUIT:      fprintf(clientlog,"<^C>");break;
            case SUBMIT:    fprintf(clientlog,"<CR>"); break;
            case TAB:       fprintf(clientlog,"<TAB>"); break;
            case RTE_SYNCH_CHARACTER:
                        fprintf(clientlog,"<^A>"); break;
            default:        fprintf(clientlog,"%c",screen[char_ct]);
        }
    }
    fprintf(clientlog,"\n");
    fflush(clientlog);
}
```

monitor.c

```
/
*****************************************************************************/
/*
**   monitor.c — All functions for Tuxedo call and return
*/
#include <stdio.h>
#include <stdarg.h>
#include “tpcc.h”
#include “monitor.h”
#ifndef DEBUG_LOOPBACK
#include <atmi.h>
#endif
#include “screen.e”
#include “infdatabuf.h”
struct PMT_DATA  pmt_data , *pmt_dataptr = &pmt_data;
struct ORDS_DATA ords_data, *ords_dataptr = &ords_data;
struct DVRY_DATA dvry_data, *dvry_dataptr = &dvry_data;
struct STKL_DATA stkl_data, *stkl_dataptr = &stkl_data;
struct NEWO_DATA newo_data, *newo_dataptr = &newo_data;




int Snd_Txn_To_Monitor(int txn_type)
{

    int status;

    void copy_NO_data(struct NEWO_DATA *, struct
io_neworder *);
    void copy_PT_data(struct PMT_DATA *, struct
io_payment*);
    void copy_OS_data(struct ORDS_DATA *, struct io_ordstat
*);
    void copy_DY_data(struct DVRY_DATA *, struct io_delivery
*);
    void copy_SL_data(struct STKL_DATA *, struct io_stocklev
*);

#if DEBUG
    Clog(“DBG:  In Snd_Txn_To_Monitor\n”);
    print_input_data(txn_type);
#endif

#ifdef DEBUG_LOOPBACK

    switch (txn_type) {
    case NEWORDER:
        LOOP_neworder_output();
        break;
    case PAYMENT:
        LOOP_payment_output();
        break;
    case ORDSTAT:
        LOOP_ordstat_output();
        break;
    case DELIVERY:
        LOOP_delivery_output();
        break;
    case STOCKLEV:
        LOOP_stocklev_output();
        break;
    }

    return 0;
#else
/*
    memcpy(Tpmbuf, (char *)ip,
        ilen);
*/

    switch (txn_type) {
    case NEWORDER:
        copy_NO_data(newo_dataptr,iNO);
        status = tpcall(“NEWO_SVC”, (char *) newo_dataptr,
            sizeof(struct NEWO_DATA), (char **) &newo_dataptr,
&olen, 0 );
        if (status == -1 && tperrno == TPESVCFAIL) {
            Clog(“ERR: Tuxedo tpcall(%s) failed \n\t%s”,
                svc_names[txn_type],tpstrerror(tperrno));
            return(TPM_ERROR);
        }
        else return(tpurcode);
        break;

    case PAYMENT:
        copy_PT_data(pmt_dataptr,iPT);
        status = tpcall(“PMT_SVC”, (char *) pmt_dataptr,
            sizeof(struct PMT_DATA), (char **) &pmt_dataptr,
&olen, 0 );
        if (status == -1 && tperrno == TPESVCFAIL) {
            Clog(“ERR: Tuxedo tpcall(%s) failed \n\t%s”,
                svc_names[txn_type],tpstrerror(tperrno));
            return(TPM_ERROR);
        }
        else return(tpurcode);
        break;

    case ORDSTAT:
        copy_OS_data(ords_dataptr,iOS);
        status = tpcall(“ORDS_SVC”, (char *) ords_dataptr,
            sizeof(struct ORDS_DATA), (char **) &ords_dataptr,
&olen, 0 );
        if (status == -1 && tperrno == TPESVCFAIL) {
            Clog(“ERR: Tuxedo tpcall(%s) failed \n\t%s”,
                svc_names[txn_type],tpstrerror(tperrno));
            return(TPM_ERROR);
        }
        else return(tpurcode);
        break;

    case DELIVERY:
        copy_DY_data(dvry_dataptr,iDY);
        status = tpacall(“DVRY_SVC”, (char *) dvry_dataptr,
            sizeof(struct DVRY_DATA), TPNOREPLY );
        if (status == -1 && tperrno == TPESVCFAIL) {
            Clog(“ERR: Tuxedo tpcall(%s) failed \n\t%s”,
                svc_names[txn_type],tpstrerror(tperrno));
            return(TPM_ERROR);
        }
        else return(tpurcode);
        break;

    case STOCKLEV:
        copy_SL_data(stkl_dataptr,iSL);
        status = tpcall(“STKL_SVC”, (char *) stkl_dataptr,
            sizeof(struct STKL_DATA), (char **) &stkl_dataptr,
&olen, 0 );
        if (status == -1 && tperrno == TPESVCFAIL) {
```

```
            Clog("ERR: Tuxedo tpcall(%s) failed \n\t%s",
                svc_names[txn_type],tpstrerror(tperrno));
            return(TPM_ERROR);
        }
        else return(tpurcode);
        break;

    }
/*
    if (txn_type == DELIVERY) {
        status = tpacall(svc_names[txn_type], Tpmbuf, ilen,
TPNOREPLY);
        if (status == -1 && tperrno == TPESVCFAIL) {
            Clog("ERR: Tuxedo tpcall(%s) failed \n\t%s",
                svc_names[txn_type],tpstrerror(tperrno));
            return(TPM_ERROR);
        }
    return;
    }
    else {
        status = tpcall(svc_names[txn_type], Tpmbuf, ilen,
&Tpmbuf, &olen,0);
        Clog("status=%d, tperrno=%d\n",status,tperrno);
        if (status == -1 && tperrno == TPESVCFAIL) {
            Clog("ERR: Tuxedo tpcall(%s) failed \n\t%s",
                svc_names[txn_type],tpstrerror(tperrno));
            return(TPM_ERROR);
        }


        return(tpurcode);
    }
*/
#endif /*ifdef DEBUG_LOOPBACK*/
}


int Init_Monitor ()
{
    char *text;

    ilen = sizeof(struct io_tpcc);
    olen = sizeof(struct io_tpcc);

#ifndef DEBUG_LOOPBACK

    if (tpinit(NULL) == -1) {
        tpmerror("tpinit",tperrno);
        return -1;
    }

/*
    if ((Tpmbuf=tpalloc("CARRAY",NULL,ilen)) == NULL) {
        tpmerror("tpalloc",tperrno);
        return (-1);
    }
*/
    if ((newo_dataptr =
        (struct NEWO_DATA *) tpalloc("CARRAY", NULL,
                                sizeof(struct NEWO_DATA)) ) ==
NULL ) {
        fprintf( stderr,
            "User-Id: %d failed at buffer allocation request for New-
Order.\n",
                user_id );
        return(1);
    }

    if ((pmt_dataptr =
        (struct PMT_DATA *) tpalloc("CARRAY", NULL,
                                sizeof(struct PMT_DATA)) ) == NULL )
{
        fprintf( stderr,
            "User-Id: %d failed at buffer allocation request for
payment.\n",
                user_id );
        return(1);

    }

    if ((ords_dataptr =
        (struct ORDS_DATA *) tpalloc("CARRAY", NULL,
                                sizeof(struct ORDS_DATA)) ) ==
NULL ) {
        fprintf( stderr,
        "User-Id: %d failed at buffer allocation request for Order-
Status.\n",
                user_id );
        return( 1 );
    }

    if ((dvry_dataptr =
        (struct DVRY_DATA *) tpalloc("CARRAY", NULL,
                                sizeof(struct DVRY_DATA)) ) ==
NULL ) {
        fprintf(stderr,
            "User-Id: %d failed at buffer allocation request for
Delivery.\n",
                user_id);
        return(1);
    }

    if ((stkl_dataptr =
        (struct STKL_DATA *) tpalloc("CARRAY", NULL,
                                sizeof(struct STKL_DATA)) ) == NULL
) {
        fprintf( stderr,
        "User-Id: %d failed at buffer allocation request for stock-
level.\n",
                user_id );
        return(1);
    }

    newo_dataptr->w_id = w_id;
    newo_dataptr->header.dtype = THEIR_NEW_ORDER;
    pmt_dataptr->w_id = w_id;
    pmt_dataptr->header.dtype = THEIR_PAYMENT;
    ords_dataptr->w_id = w_id;
    ords_dataptr->header.dtype = THEIR_ORDER_STATUS;
    stkl_dataptr->w_id      = w_id;
    stkl_dataptr->stkl_d_id = d_id;
    stkl_dataptr->header.dtype = THEIR_STOCK_LEVEL;
/*
    dvry_dataptr->startq_sec = startq_t.tv_sec;
    dvry_dataptr->startq_usec = startq_t.tv_usec;
*/
    dvry_dataptr->w_id     = w_id;
    dvry_dataptr->header.dtype  = DELIVERY;



#else
```

```
    if ((Tpmbuf = (char *)malloc(ilen)) == NULL) {
        return (-1);
    }

#endif
    return (NULL);
}

Rundown_Monitor()
{

int status;

#ifndef DEBUG_LOOPBACK

    tpfree(Tpmbuf);

    status = tpterm();
#ifdef DEBUG
    Clog("terminated Tuxedo connection with status
%d\n",status);
#endif
#endif
}

#ifndef DEBUG_LOOPBACK
tpmerror(char *service_called,int errnum)
{
    char errmsg[256];

    fprintf(stderr,"TUXEDO:  Failed %s with error:  %s\n",
        service_called,tpstrerror(errnum));
    fprintf(stderr,"\n");
}
#endif

#ifdef DEBUG
print_input_data(int type)
{

    int i;
    time_t the_time;

    the_time = time(&the_time);
    Clog("DBG:==============TIME: %s
==============\n",ctime(&the_time));

    switch (type) {

    case NEWORDER:
        Clog("DBG: NEWORDER INPUTS at
%s\n",ctime(&the_time));
        Clog("DBG: w_id: %d, d_id: %d, c_id: %d o_ol_cnt:
%d\n",
            iNO->w_id,iNO->d_id, iNO->c_id, iNO->o_ol_cnt);
        for (i=0; i < iNO->o_ol_cnt; i++)
        Clog("DBG: ol_i_id: %d, ol_supply_w_id: %d, ol_quantity:
%d\n",
              iNO->o_ol[i].ol_i_id,iNO->o_ol[i].ol_supply_w_id,
              iNO->o_ol[i].ol_quantity);
        break;
    case PAYMENT:
            Clog("DBG: PAYMENT INPUTS at
%s\n",ctime(&the_time));
            Clog("DBG: byname: %d, w_id: %d, d_id: %d\n", iPT-
>byname,
                        iPT->w_id,iPT->d_id);
        Clog("DBG: c_last: %s ",iPT->c_last);
        Clog(" c_id: %d",iPT->c_id);
        Clog(" c_w_id: %d, c_d_id: %d\n",iPT->c_w_id,iPT-
>c_d_id);
        Clog("DBG: h_amount: %f\n",iPT->h_amount);
        break;
    case ORDSTAT:
        Clog("DBG: ORDER STATUS INPUTS at
%s\n",ctime(&the_time));
        Clog("DBG: w_id: %d, d_id: %d\n",iOS->w_id,iOS->d_id);
        Clog("DBG: byname: %d, c_id: %d, c_last: %s\n",
            iOS->byname,iOS->c_id,iOS->c_last);
        break;
    case DELIVERY:
        Clog("DBG: DELIVERY INPUTS at
%s\n",ctime(&the_time));
        Clog("DBG: w_id: %d, o_carrier_id: %d\n",iDY->w_id,iDY-
>o_carrier_id);
        break;
    case STOCKLEV:
        Clog("DBG: STOCK LEVEL INPUTS at
%s\n",ctime(&the_time));
        Clog("DBG: w_id: %d, d_id: %d, threshold: %d\n",iSL-
>w_id,iSL->d_id,
            iSL->threshold);
        break;
    other:
        Clog("DBG: Txn_type = %d is illegal  at
%s\n",type,ctime(&the_time));
    }
    return;
}

#endif  /* ifdef DEBUG */

void copy_NO_data(struct NEWO_DATA *theirs, struct
io_neworder *mine) {

   int i;

   theirs->w_id = (short) mine->w_id;
   theirs->d_id = (short) mine->d_id;
   theirs->c_id = mine->c_id;
   theirs->o_ol_cnt = (short) mine->o_ol_cnt;
   theirs->o_all_local = (short) mine->o_all_local;

   for (i = 0; i < mine->o_ol_cnt; i++) {
      theirs->ol_table[i].ol_supply_w_id = (short) mine-
>o_ol[i].ol_supply_w_id;
      theirs->ol_table[i].ol_quantity = (short) mine-
>o_ol[i].ol_quantity;
      theirs->ol_table[i].ol_i_id = mine->o_ol[i].ol_i_id;
   }
}

void copy_PT_data(struct PMT_DATA *theirs, struct
io_payment *mine) {
   theirs->w_id = (short) mine->w_id;
   theirs->d_id = (short) mine->d_id;
   theirs->c_id = mine->c_id;
   theirs->c_w_id = (short) mine->c_w_id;
   theirs->c_d_id = (short) mine->c_d_id;
   theirs->byname = (short)mine->byname;
   theirs->h_amount = mine->h_amount;
    strcpy(theirs->c_last,mine->c_last);
}
void copy_OS_data(struct ORDS_DATA *theirs, struct
```

```
io_ordstat *mine) {
   theirs->w_id = (short) mine->w_id;
   theirs->d_id = (short) mine->d_id;
   theirs->c_id = mine->c_id;
   theirs->byname = (short)mine->byname;
    strcpy(theirs->c_last,mine->c_last);
#ifdef DEBUG
Clog("DBG: after copy Orderstatus INPUTS \n");
Clog("DBG: w_id: %d, d_id: %d, c_id: %d, byname: %d\n",
   theirs->w_id,theirs->d_id,theirs->c_id,theirs->byname);
#endif
}
void copy_DY_data(struct DVRY_DATA *theirs, struct
io_delivery *mine) {
   theirs->w_id = (short) mine->w_id;
   theirs->carrier_id = (short) mine->o_carrier_id;
   theirs->startq_sec = mine->startq_sec;
   theirs->startq_usec = mine->startq_usec;
#ifdef DEBUG
Clog("DBG: after copy in Delivery\n");
Clog("DBG: w_id=%d carrier=%d startq_sec=%d
startq_used=%d\n",
    theirs->w_id, theirs->carrier_id, theirs->startq_sec, theirs-
>startq_usec);
#endif

}
void copy_SL_data(struct STKL_DATA *theirs, struct
io_stocklev *mine) {
   theirs->w_id = (short) mine->w_id;
   theirs->stkl_d_id = (short) mine->d_id;
   theirs->threshold = mine->threshold;
}
```

monitor.e

```
/
*******************************************************************************
*****************************  monitor.e
************************************
*******************************************************************************/
/*
**  monitor.e — external definitions for monitor calls — includes
some
**                      handy #defines for access to the returned
monitor data.
*/
extern int Snd_Txn_To_Monitor(int);
extern int Init_Monitor();
extern void Rundown_Monitor();

extern struct io_tpcc IO_Data;

extern char *Tpmbuf;



/*  Convenient definitions for accessing inputs and outputs */
#define oNO  (&((struct io_tpcc *) Tpmbuf)->info.neworder)
#define oPT  (&((struct io_tpcc *) Tpmbuf)->info.payment)
#define oOS  (&((struct io_tpcc *) Tpmbuf)->info.ordstat)
#define oDY  (&((struct io_tpcc *) Tpmbuf)->info.delivery)
#define oSL  (&((struct io_tpcc *) Tpmbuf)->info.stocklev)

#define ip (&IO_Data)

#define iNO (&IO_Data.info.neworder)
#define iPT (&IO_Data.info.payment)
#define iOS (&IO_Data.info.ordstat)
#define iDY (&IO_Data.info.delivery)
#define iSL (&IO_Data.info.stocklev)
```

monitor.h

```
/
*******************************************************************************
*****************************                          monitor.h
************************************
*******************************************************************************/
/*
** monitor.h — All Tuxedo definitions and storage
**
*/



   long ilen;
   long olen;


char *svc_names[] = { "NEWORDER_SVC",
                      "PAYMENT_SVC",
                      "ORDSTAT_SVC",
                      "DELIVERY_SVC",
                      "STOCKLEV_SVC" };


/* The actual IO storage is allocated here as IO_Data.   The
routines in
screen.c fill this up.  The data is copied into Tpmbuf to send to
tuxedo.
Data is returned in Tpmbuf.  Some #defines are set up here for
easy
access/naming of the input and output areas
*/


struct io_tpcc IO_Data;
char *Tpmbuf;

extern void Clog(char *, ...);

#define oNO  (&((struct io_tpcc *) Tpmbuf)->info.neworder)
#define oPT  (&((struct io_tpcc *) Tpmbuf)->info.payment)
#define oOS  (&((struct io_tpcc *) Tpmbuf)->info.ordstat)
#define oDY  (&((struct io_tpcc *) Tpmbuf)->info.delivery)
#define oSL  (&((struct io_tpcc *) Tpmbuf)->info.stocklev)

#define ip (&IO_Data)
#define iNO (&IO_Data.info.neworder)
#define iPT (&IO_Data.info.payment)
#define iOS (&IO_Data.info.ordstat)
#define iDY (&IO_Data.info.delivery)
```

```
#define iSL (&IO_Data.info.stocklev)

/*

#define ip  (struct io_tpcc *) Tpmbuf
#define iNO (&((struct io_tpcc *) Tpmbuf)->info.neworder)
#define iPT (&((struct io_tpcc *) Tpmbuf)->info.payment)
#define iOS (&((struct io_tpcc *) Tpmbuf)->info.ordstat)
#define iDY (&((struct io_tpcc *) Tpmbuf)->info.delivery)
#define iSL (&((struct io_tpcc *) Tpmbuf)->info.stocklev)
*/
```

screen.c

```
/
*******************************************************************************
****************************** screen.c
************************************
*******************************************************************************/
/*
**  screen.c:  Contains all routines for the TPC-C screen
display, input and
**             output.
**
*******************************************************************************

                    COPYRIGHT (C) 1994 BY
                DIGITAL EQUIPMENT CORPORATION,
MAYNARD
                 MASSACHUSETTS.  ALL RIGHTS RESERVED.

  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND
MAY BE USED AND COPIED
  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH
LICENSE AND WITH THE INCLUSION
  OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE
OR ANY OTHER COPIES
  THEREOF MAY NOT BE PROVIDED OR OTHERWISE
MADE AVAILABLE TO ANY OTHER
  PERSON.  NO TITLE TO AND OWNERSHIP OF THE
SOFTWARE IS HEREBY TRANSFERRED.

  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO
CHANGE WITHOUT NOTICE AND
  SHOULD NOT BE CONSTRUED AS A COMMITMENT BY
DIGITAL EQUIPMENT CORPORATION.

  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE
OR RELIABILITY OF ITS
  SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY
DIGITAL.

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

 **/

#include <stdio.h>
#include <sys/termio.h>
#include <stdlib.h>

#include <sys/time.h>
#include <time.h>

#include "tpcc.h"
#include "screen.e"
#include "screen.h"
#include "monitor.e"

#include "infdatabuf.h"


extern struct NEWO_DATA *newo_dataptr;
extern struct PMT_DATA *pmt_dataptr;
extern struct ORDS_DATA *ords_dataptr;
extern struct DVRY_DATA *dvry_dataptr;
extern struct STKL_DATA *stkl_dataptr;

extern void Clog(char *, ...);
extern void SCREENlog(int ,char *);
#define MAXLINE 256

void setraw()  /** put terminals into rawmode  **/
{
    extern struct tbufsave;
    struct termio tbuf;
    int status;
    if (ioctl(tty_in, TCGETA, &tbuf) == -1)
       syserr("ioctl_ERROR#1 - getting the original input term
setting error");
    tbufsave = tbuf;
    tbuf.c_iflag &= ~(INLCR | ICRNL | IUCLC | ISTRIP | IXON |
BRKINT);
    tbuf.c_oflag &= ~OPOST;
    tbuf.c_lflag &= ~(ICANON | ISIG | ECHO);
    tbuf.c_cc[VMIN] = LEAVE_SCREEN_MIN;
    tbuf.c_cc[VTIME] = LEAVE_SCREEN_TIMEOUT;
    if (ioctl(tty_out, TCSETAF, &tbuf) == -2)
       syserr("ioctl_ERROR#2 - setting raw mode for STDIN
error");
}

void restore_terminal() /** restore terminal flags **/
{
    extern struct tbufsave;
    struct termio tbuf;
    int status;

    if (ioctl(tty_out, TCSETAF, &tbufsave) == -1)
        syserr("ioctl_ERROR#3 - restoring original input terminal
settings error");
    tbuf = tbufsave;

    if (ioctl(tty_out, TCSETAF, &tbuf) == -1)
        syserr("ioctl_ERROR#4 - Forcing the original settings
back for STDIN error");
    if (ioctl(tty_out, TCSETAF, &tbuf) == -1)
        syserr("ioctl_ERROR#5 - Forcing the original settings
back for STDOUT error");
}

int Get_Menu_Input()
{
    int c, read_count;
    static char inbuf[2] = "\0\0";
    int i = 0;
    read_count = read(tty_in,inbuf,1);
    if (read_count == 0) syserr ("TTY lost connection");

    if (inbuf[0] == QUIT)
        return 9;
    c = atoi(inbuf);
```

```
    return c;
/* Code for 2 character
    read_count = read(tty_in,inbuf,2);
    if (read_count == 0) syserr ("TTY lost connection");

    if (inbuf[0] == QUIT)
        return 9;
    c = atoi(inbuf);

    if (read_count != 2) read(tty_in,inbuf,1);
    return c;
*/
}

int Get_Form_Data(int txn_type)
{
    BOOLEAN done=FALSE;
    int i, returned_key;
    io_elem *ioptr;
    int last_input;



    BOOLEAN check_neworder_inputs(int *);
    BOOLEAN check_payment_inputs(int *);
    BOOLEAN check_ordstat_inputs(int *);
    BOOLEAN check_delivery_inputs(int *);
    BOOLEAN check_stocklev_inputs(int *);


    BOOLEAN (*p_check_function[])() = {
        &check_neworder_inputs,
        &check_payment_inputs,
        &check_ordstat_inputs,
        &check_delivery_inputs,
        &check_stocklev_inputs
    };

     memset(ip,'\0',sizeof(struct io_tpcc));
     memset(orig_ol,'\0',sizeof(orig_ol));
    int_h_amount = 0;
    last_input = Forms[txn_type].num_input_elems -1;


    i = 0;
    while (done == FALSE)
    {
        ioptr = &Forms[txn_type].input_elems[i];
        if (i == 5 && txn_type == PAYMENT) payment_input =
TRUE;
        if (i != 5 && txn_type == PAYMENT) payment_input =
FALSE;
        returned_key = (ioptr->fptr)(ioptr->x,ioptr->y,ioptr->len,
             ioptr->flags,ioptr->dptr);
#ifdef DEBUG
Clog("returned_key %o\n",returned_key);
#endif
        switch (returned_key)
        {
            case BACKTAB:
                if (i == 0) i = last_input ;
                else i—;
                break;
            case TAB:
                if (i == last_input) i = 0;
                else i++;
                break;
            case QUIT:
                done = TRUE;
                break;
            case LF:
            case SUBMIT:
                payment_input = FALSE;
                done = (p_check_function[txn_type])(&i);
                ip->type = txn_type;
                break;
        }
    }
    return returned_key;

}

BOOLEAN check_neworder_inputs(int *pos)
{
    BOOLEAN done = FALSE;
    struct io_order_line *real_ol_ptr;
    struct orig_order_line_struct *orig_ol_ptr;
    int i;


    iNO->w_id = w_id;
    if (iNO->d_id <= 0) {
        *pos = 0;
        message = TRUE;
        PAINTSCR(MANDATORY_MSG);

    }
    else if (iNO->c_id <= 0) {
        *pos = 1;
        message = TRUE;
        PAINTSCR(MANDATORY_MSG);
    }
    else  {

        orig_ol_ptr = orig_ol;
        real_ol_ptr = iNO->o_ol;

        iNO->o_all_local = 1;

        for (i = 0; i < MAX_OL; i++,orig_ol_ptr++){

                                /* Is there data on this line? */
            if (orig_ol_ptr->o_ol.ol_i_id || orig_ol_ptr-
>o_ol.ol_supply_w_id
                    || orig_ol_ptr->o_ol.ol_quantity) {
                                /* and is that data complete */
                if (orig_ol_ptr->o_ol.ol_i_id && orig_ol_ptr-
>o_ol.ol_supply_w_id
                     && orig_ol_ptr->o_ol.ol_quantity)        {
                                /* if fine, then copy to io struct */
                    iNO->o_ol_cnt++;
                    memcpy(real_ol_ptr,orig_ol_ptr,
                                  sizeof(struct io_order_line));
                    if (iNO->w_id != real_ol_ptr->ol_supply_w_id)
                        iNO->o_all_local = 0;
                    real_ol_ptr++;
                    orig_ol_ptr->ol_loc= iNO->o_ol_cnt;
                }                /* if not fine, go back */
                else {
                    *pos = 2 + 3*i;
                    PAINTSCR(INCOMPLINE_MSG);
                    message = TRUE;
                    iNO->o_ol_cnt = 0;
                    iNO->o_all_local = 1;
```

```
                return FALSE;
              }
            }
          }
          if (!iNO->o_ol_cnt) {
            *pos = 2;
            PAINTSCR(MANDATORY_MSG);
            message = TRUE;
            iNO->o_ol_cnt=0;
             memset(orig_ol,'\0',sizeof(struct
orig_order_line_struct));
            return FALSE;
          }

          done = TRUE;
#ifdef DEBUG
Clog("Recieved proper Neworder inputs\n");
#endif
    }

    return done;
}



BOOLEAN check_payment_inputs(int *pos)
{

    BOOLEAN done=FALSE;
    iPT->w_id = w_id;
    if (iPT->d_id <= 0) {
        *pos = 0;
        message = TRUE;
        PAINTSCR(MANDATORY_MSG);
    }
    else if (iPT->c_w_id <= 0) {
        *pos = 3;
        message = TRUE;
              PAINTSCR(MANDATORY_MSG);
    }
    else if (iPT->c_d_id <= 0) {
        *pos = 4;
        message = TRUE;
        PAINTSCR(MANDATORY_MSG);
    }
    else if (int_h_amount <= 0) {
              *pos = 5;
        message = TRUE;
        PAINTSCR(MANDATORY_MSG);
    }
    else if (iPT->c_id <= 0) {
        if (iPT->c_last[0] == '\0') {
            message = TRUE;
                 PAINTSCR(ID_OR_LAST_MSG);
            *pos = 1;
              }
              else {
            iPT->byname = TRUE;
                 done = TRUE;
        }
    }
    else
        done = TRUE;


    byname = iPT->byname;
    iPT->h_amount = int_h_amount/100.0;

    return done;
}



BOOLEAN check_ordstat_inputs (int *pos)

{
    BOOLEAN done = FALSE;
    iOS->w_id = w_id;
    if (iOS->d_id <= 0) {
        *pos = 0;
        message = TRUE;
        PAINTSCR(MANDATORY_MSG);
    }
    else if (iOS->c_id <= 0) {
        if (iOS->c_last[0] == '\0'){
            message = TRUE;
            PAINTSCR(ID_OR_LAST_MSG);
            *pos = 1;
        }
        else {
            iOS->byname = TRUE;
            done = TRUE;
        }
    }
    else
        done = TRUE;


    byname = iOS->byname;

    return done;
}

BOOLEAN check_delivery_inputs (int *pos)
{

    BOOLEAN done = FALSE;
    struct  timeval startq_t;
    struct  timezone startq_z;  /* for delivery startq values */


    iDY->w_id = w_id;
    if (iDY->o_carrier_id <= 0) {
        message = TRUE;
        PAINTSCR(MANDATORY_MSG);
    }
    else {
/*
        time(&iDY->queue_time);
*/
        gettimeofday(&startq_t, &startq_z);
        iDY->startq_sec = startq_t.tv_sec;
        iDY->startq_usec = startq_t.tv_usec;

       done = TRUE;

    }

    return done;
}

BOOLEAN check_stocklev_inputs(int *pos)
{
    BOOLEAN done = FALSE;
```

```
    iSL->w_id = w_id;
    iSL->d_id = d_id;
    if (iSL->threshold <= 0) {
        message = TRUE;
        PAINTSCR(MANDATORY_MSG);
    }
    else done = TRUE;

    return done;
}


void Init_Screen()
{
    int i;
    char    buf[128];
    void setup_io_elems();

    setraw();
    for (i=0; i < MAX_FORMS; i++)
        setup_screen_buffer(&Forms[i],i);
    setup_io_elems();
    CLRSCN(buf);
    PAINTSCR(buf);
}

void Restore_Screen()
{
    restore_terminal();
}

void Paint_Screen(int screen_num)
{
    if (PAINTSCR(Forms[screen_num].blank_form) == -1)
        syserr("Can't write out form");
}

void Send_Menu()
{
    if (PAINTSCR(menu_buf) == -1)
        syserr("Can't send menu");
}

void setup_io_elems() {
    io_elem *p;
    int i;


    p = Forms[NEWORDER].input_elems;
    p++->dptr = &iNO->d_id;
    p++->dptr = &iNO->c_id;
    for (i=0; i < 15; i++){
        p++->dptr = &orig_ol[i].o_ol.ol_supply_w_id;
        p++->dptr = &orig_ol[i].o_ol.ol_i_id;
        p++->dptr = &orig_ol[i].o_ol.ol_quantity;
    }

    p = Forms[PAYMENT].input_elems;
    p++->dptr = &iPT->d_id;
    p++->dptr = &iPT->c_id;
    p++->dptr = (int *) &iPT->c_last[0];
    p++->dptr = &iPT->c_w_id;
    p++->dptr = &iPT->c_d_id;
    p->dptr = &int_h_amount;

    p = Forms[ORDSTAT].input_elems;
    p++->dptr = &iOS->d_id;
    p++->dptr = &iOS->c_id;
    p->dptr = (int *) &iOS->c_last[0];

    p = Forms[DELIVERY].input_elems;
    p->dptr = &iDY->o_carrier_id;

    p = Forms[STOCKLEV].input_elems;
    p->dptr = &iSL->threshold;

}

int setup_screen_buffer(struct form_info *form_ptr,int txn_type
){

    FILE    *ifile;
    text_elem *tbuf;
    char    *bufp;
    int     ct;
    char    blanks[] = "                                ";
    char    input_display_buf[64];
    char    fname[MAXLINE];
    io_elem *io_ptr;

    bufp = form_ptr->blank_form;
    bufp += CLRSCN(bufp);

    tbuf = form_ptr->tp;

    while ( tbuf->text ) {
       bufp += DISPLAY(bufp,tbuf->y,tbuf->x,tbuf->text);
       tbuf++;
    }

    bufp += SWITCH_TO_UNDERL(bufp);
    ct = 0;
    for (io_ptr=form_ptr->input_elems ; io_ptr->y != 999;
io_ptr++) {
        strncpy(input_display_buf, blanks, io_ptr->len);
        input_display_buf[io_ptr->len]='\0';
        bufp += DISPLAY(bufp,io_ptr->x,io_ptr-
>y,input_display_buf);
        ct++;
    }

    form_ptr->num_input_elems = ct;

    bufp += SWITCH_TO_NORMAL(bufp);

    if (txn_type == PAYMENT)
        bufp += DISPLAY_INT(bufp,4,12,4,w_id);
    else
        bufp += DISPLAY_INT(bufp,4,12,2,w_id);
    if (txn_type == STOCKLEV)
        bufp += DISPLAY_INT(bufp,2,29,2,d_id);
    bufp += SWITCH_TO_UNDERL(bufp);
    *bufp++ = '\1';

    *bufp = '\0';
}



int read_integer(col, row, size, flags, data)
int col, row, size, flags, *data;

/** Function to read in integer data from the screen.
```

```
    col   - first column position of the data field on the screen.
    row   - row position of the data field on the screen.
    size  - the length of the data field in number of characters to
read in.
    flags - boolean flag that was used to identify the mandatory
data fields.
    data  - pointer to the actual location where the returning
value should be stored in.

    This function reads in the input data until either a TAB,
BACKTAB, LINEFEED,
    SUBMIT character is entered or the maximum number of
characters have been
    entered into the data field.

    The data is read as character and converted to an integer
before being
    stored in the appropriate location.

 **/
{
    BOOLEAN exit_read_function=FALSE,
previous_data_exists=FALSE;
    int return_status=TAB, bytes_read=0, i=0, j=0,
k=0,size1=0,cur_col=col;
    char *bufp, *tempbuf,temp[50],  blanks[]="        ";
    float q;
    char bsspbs[]="\033[D \033[D", erase_field[20];
    static char screen_buf[200];

    strncpy(temp, "\0", 20);
    screen_buf[0] = '\0';
    bufp = screen_buf;                          /* Position cursor at
start of field */
    bufp += GOTOXY(bufp,col+size-1,row);
    PAINTSCR(screen_buf);

    bufp = screen_buf;
    size1=size;

    if (*data > 0)
        previous_data_exists = TRUE;
    while (exit_read_function == FALSE)
    {
/*
   Below we read from standard input into the array curbuf.
   curbuf_read is the pointer to the array curbuf indicating the
position
   upto which the curbuf has been parsed.
   curbuf_consumed is the number of elements in the buffer
temp that holds the
   array that is to be displayed.
   Elements of curbuf_consumed is selectively copied from
curbuf

   Note:read_count is the total number of characters in the
buffer
   curbuf. curbuf_read is always less than or equal to
read_count.
*/
        if (curbuf_read == read_count || curbuf_read == 0)
        {
        curbuf_read = 0;
        read_count = read(tty_in, curbuf, sizeof(curbuf));
        if (read_count == 0) syserr ("TTY lost connection");
        }
/* BOOLEAN message prevents unnecessary display of
warning messages */
        if (message == TRUE)
        {
            bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW, ERASE_MSG);
            message=FALSE;
        }
        if (previous_data_exists == TRUE)
        {
            if (curbuf[curbuf_read] == DELETE)
            {
                previous_data_exists = FALSE;
                strncpy(erase_field, blanks, size);
                erase_field[size] = '\0';
                bufp += DISPLAY(bufp, col, row, erase_field);
                bufp += GOTOXY(bufp,col+size-1,row);
            }
            else
            {
                if (curbuf[curbuf_read] < '0' || curbuf[curbuf_read] >
'9')
                {
                    exit_read_function = TRUE;
                    previous_data_exists = FALSE;
                    return_status = curbuf[curbuf_read];
                    curbuf[curbuf_read]='\0';
                }
                else
                {
                    previous_data_exists = FALSE;
                    strncpy(erase_field, blanks, size);
                    erase_field[size] = '\0';
                    bufp += DISPLAY(bufp, col, row, erase_field);
/*
                    bufp = screen_buf;
*/
                }
            }
        }  /* if previous_data_exists */

        while ((curbuf_read < read_count) && (exit_read_function
== FALSE))
        {
#ifdef DEBUG
int xx;
Clog("Initial: Current character %o, curbuf_read = %d
read_count=%d\n ",
                curbuf[curbuf_read],curbuf_read,read_count);
/*
for (xx = 0; xx <= read_count; xx++)
       Clog("xx = %d, curbuf[xx] = %o\n",xx,curbuf[xx]);
*/
#endif
            /* intermediate variable size1 for cases when floating
point
               field whose size is less than actual size by 1 because
of
               decimal.
             */
            if (payment_input == TRUE) size1=size-1;
                                                /* Test for integer   */
            if (curbuf[curbuf_read] >= '0' && curbuf[curbuf_read] <=
'9')
            {
                                                /* Consume all integers
in buffer */
                for ( ;curbuf[curbuf_read] >= '0'
```

```
                        && curbuf[curbuf_read] <= '9'
;curbuf_read++)
/*
#ifdef DEBUG
Clog("Current character %o, curbuf_read = %d
read_count=%d\n ",
            curbuf[curbuf_read],curbuf_read,read_count);
#endif
*/
               {
                 /* below we fill up temp making sure the size limit
                    is not exceeded */
                 if (curbuf_consumed < size1)
                        {
                          temp[curbuf_consumed] =
curbuf[curbuf_read];
                          curbuf_consumed++;
                        }
                 /* number of elements typed in is more than the
size of
                    the field */
                 else
                     OVERFLOW = TRUE;
                 /* ensure the character is removed after it is read
*/
               curbuf[curbuf_read] = '\0';

               } /*  end of for curbuf is legitimate number */

               temp[curbuf_consumed] = '\0';  /* terminate temp
string */

                if (payment_input == TRUE) /* floating point field */
                {
                 /* convert the ascii to float */
                 q = (atof(temp))/100;
                 if (curbuf_consumed < 3)
                 bufp += DISPLAY_FLOAT(bufp, 2,(col+size-4),
row, q);
                 else
                 bufp += DISPLAY_FLOAT(bufp, 2,(col+size-
curbuf_consumed-1),
                                 row, q);
               }
                else
               {
                 if (curbuf_consumed < size+1)
                 bufp += DISPLAY(bufp,(col + size -
curbuf_consumed),row,
                                 temp);
                 return_status = curbuf[curbuf_read];
                 cur_col++;
                 }
          }  /* if curbuf[] between "0" and "9" */

                                    /* if not integer, then test for
movement character */
          else if (   curbuf[curbuf_read] == TAB
                   || curbuf[curbuf_read] == LF
                   || curbuf[curbuf_read] == BACKTAB
                   || curbuf[curbuf_read] == SUBMIT)
          {
                 if (message == TRUE)
                  {
                    bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
                          ERASE_MSG);
                    message=FALSE;
                  }
               temp[curbuf_consumed] = '\0';
               *data = atoi(temp);
               exit_read_function = TRUE;
               return_status = curbuf[curbuf_read];
               curbuf[curbuf_read] = '\0';
               curbuf_read++;
               curbuf_consumed = 0;
          }  /* if curbuf[] a movement character */

                        /* if not integer of movement, test for
DELETE */
          else if (curbuf[curbuf_read] == DELETE)
          {
             if (payment_input == TRUE) /* for floating point field
*/
               {
                 if (curbuf_consumed != 0) curbuf_consumed—;
                 if (message == TRUE)
                  {
                    bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
                                 ERASE_MSG);
                   message=FALSE;
                  }
                 OVERFLOW = FALSE;
                 PAINTSCR(screen_buf);
                 temp[curbuf_consumed] = '\0';
                 q=atof(temp);
                 q=(q/100);
                 curbuf[curbuf_read] = '\0';
                 strncpy(erase_field, blanks, size);
                 erase_field[size] = '\0';
                 bufp = screen_buf;
                 bufp += DISPLAY(bufp, col, row, erase_field);
                 if (curbuf_consumed < 3)
                     bufp += DISPLAY_FLOAT(bufp, 2,
                           (col+size-4), row, q);
                 else
                     bufp += DISPLAY_FLOAT(bufp, 2,
                           (col+size-curbuf_consumed-1), row, q);
                 if (cur_col != 0) cur_col—;
                 if ( curbuf_read < 40) curbuf_read++;  /* pressed
                                              key overflow
                                              situations */
                 bufp += GOTOXY(bufp,col+size,row);
               }
             else
               {
                 if (curbuf_consumed != 0) curbuf_consumed—;
                 curbuf[curbuf_read] = '\0';
                 curbuf_read++;
                 if (message == TRUE)
                  {
                    bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
                                 ERASE_MSG);
                    message=FALSE;
                  }
                 OVERFLOW = FALSE;
                 PAINTSCR(screen_buf);
                 temp[curbuf_consumed] = '\0';
                 strncpy(erase_field, blanks, size);
                 erase_field[size] = '\0';
                 bufp = screen_buf;
                 bufp += DISPLAY(bufp, col, row, erase_field);
```

```
                bufp += DISPLAY(bufp,(col+size-
curbuf_consumed),row, temp);
                if (cur_col != 0) cur_col—;
                bufp += GOTOXY(bufp,col+size,row);
               }
        }  /* end of if DELETE */

                    /* could be a ^C */
         else if (curbuf[curbuf_read] == QUIT)
         {
                temp[0] = '\0';
                return_status = QUIT;
                curbuf[curbuf_read]='\0';
                exit_read_function = TRUE;
         }
         else /**   Any other character entered at the keyboard
...   **/
         {
            if (message == FALSE) {
                bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
                    INVALID_MSG);
#ifdef DEBUG
Clog("The invalid character was %o \n",curbuf[curbuf_read]);
#endif
                bufp += GOTOXY (bufp,col + size ,row);
                PAINTSCR(screen_buf);

                message = TRUE;
            }
            curbuf_read++;
         }
      } /**  End of the WHILE loop  **/
      if (OVERFLOW == TRUE && exit_read_function ==
FALSE)
      {
      /* if number of characters are exceeding the field limit
beep
         and warning message is necessary */
          if (message == FALSE)
          {
            bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW, EXC_FLD_LIM_MSG);
            PAINTSCR(screen_buf);
            message=TRUE;
          }
          *data = atoi(temp);
          return_status = curbuf[curbuf_read];
          curbuf[curbuf_read] = '\0';
          curbuf_read=0;
          OVERFLOW = FALSE;
      }
      else
      {
        PAINTSCR(screen_buf);
        bufp = screen_buf;
      }
   }
/* ensuring unnecessary warning messages are removed */
  if (message == TRUE)
     {
          bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW, ERASE_MSG);
           message=FALSE;
           PAINTSCR(screen_buf);
           bufp = screen_buf;
     }
    return (return_status);
}


int read_string(col, row, size, flags, data)
int col, row, size, flags;
char *data;
{
    BOOLEAN exit_read_function=FALSE,
previous_data_exists=FALSE,
            data_full=FALSE;
    int return_status=TAB, bytes_read=0, i=0, j=0, size_tot =0;
    char *bufp, temp[80];
    char blanks[]="                                   ";
    char bsspbs[]="\033[4m                \033[4m", erase_field[20];
    static char screen_buf[200];

    strncpy(temp, "\0", 20);
    curbuf_consumed = 0;
    screen_buf[0] = '\0';
    bufp = screen_buf;

    bufp += GOTOXY(bufp,col,row);       /* Goto input area */
    PAINTSCR(screen_buf);
    bufp = screen_buf;

    if ((*(char *)data) != '\0')
        previous_data_exists = TRUE;

    while(exit_read_function == FALSE)
    {
/*
   Below we read from standard input into the array curbuf.
   curbuf_read is the pointer to the array curbuf indicating the
position
   upto which the curbuf has been parsed.
   curbuf_consumed is the number of elements in the buffer
temp that holds the
   array that is to be displayed.
   Elements of curbuf_consumed is selectively copied from
curbuf

   Note:read_count is the total number of characters in the
buffer
   curbuf. curbuf_read is always less than or equal to
read_count.
*/
       if (curbuf_read == read_count)
       {
         curbuf_read = 0;
         read_count = read(tty_in, curbuf, size-size_tot);
         if (read_count == 0) syserr ("TTY lost connection");
       }
       if (message == TRUE)
       {
            bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW, ERASE_MSG);
            message=FALSE;
       }
       if (previous_data_exists == TRUE)
       {
           if (curbuf[curbuf_read] == DELETE)
           {
               previous_data_exists = FALSE;
               strncpy(erase_field, blanks, size);
               erase_field[size] = '\0';
```

```
            bufp += DISPLAY(bufp, col, row, erase_field);
            bufp += GOTOXY(bufp,col,row);
        }
        else
        {
            if (curbuf[curbuf_read] < ' ' || curbuf[curbuf_read] >
'~')
            {
                exit_read_function = TRUE;
                previous_data_exists = FALSE;
                return_status = curbuf[curbuf_read];
                curbuf[curbuf_read]='\0';
            }
            else
            {
                previous_data_exists = FALSE;
                strncpy(erase_field, blanks, size);
                erase_field[size] = '\0';
                bufp += DISPLAY(bufp, col, row, erase_field);
/*
                bufp = screen_buf;
*/
                bufp += GOTOXY(bufp,col,row);
            }
        }
    }
    while ((curbuf_read < read_count) && (exit_read_function
== FALSE))
    {
        if (curbuf[curbuf_read] >= ' ' && curbuf[curbuf_read] <=
'~')
        {   /**  if between ASCII space (040) through ~ (0176)
**/
            for( ;curbuf[curbuf_read] >= ' '
                && curbuf[curbuf_read] <= '~' ;curbuf_read++)
             {
              /* ensuring the curbuf_consumed is not more than
                 field size */
              if (curbuf_consumed < size)
                        {
                          temp[curbuf_consumed] =
curbuf[curbuf_read];
                          curbuf_consumed++;
                        }
              /* else overflow condition */
              else OVERFLOW = TRUE;
              curbuf[curbuf_read] = '\0'; /* erasing characters
already
                                             read from the buffer */
             }
            temp[curbuf_consumed] = '\0';  /* terminate temp
string */
            bufp += DISPLAY(bufp, col, row, temp);
            return_status = curbuf[curbuf_read];
        }
        else if (   curbuf[curbuf_read] == TAB
                 || curbuf[curbuf_read] == LF
                 || curbuf[curbuf_read] == BACKTAB
                 || curbuf[curbuf_read] == SUBMIT)
        {
            if (curbuf_consumed > 0)
            {
             if (message == TRUE)
                {
                    bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
                                    ERASE_MSG);
                    message=FALSE;
                }
                temp[curbuf_consumed] = '\0';
                strcpy(data, temp);
                exit_read_function = TRUE;
                return_status = curbuf[curbuf_read];
                curbuf[curbuf_read]='\0';
                curbuf_read++;
                curbuf_consumed = 0;
            }
            else
            {
             if (message == TRUE)
                {
                    bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
                                    ERASE_MSG);
                    message=FALSE;
                }
                temp[curbuf_consumed] = '\0';
                strcpy(data, temp);
                exit_read_function = TRUE;
                return_status = curbuf[curbuf_read];
                curbuf[curbuf_read]='\0';
                curbuf_read++;
            }
        }
        else if (curbuf[curbuf_read] == DELETE)
        {
            for(curbuf_read = curbuf_read;curbuf[curbuf_read]
== DELETE
                                    ;curbuf_read++)
                {

                        curbuf[curbuf_read]='\0';
                        temp[curbuf_consumed-1] = '\0';
                        if (curbuf_consumed != 0)
curbuf_consumed—;
                }
                if (curbuf_consumed >= 0)
                {
                  bufp += BLANK_UNDERLINE(bufp,col, row, "
");
                  bufp += DISPLAY(bufp, col, row, temp);
                  PAINTSCR(screen_buf);
                }
                else
                {
                  if (message == FALSE)
                  {
                    bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
                                                EXC_FLD_LIM_MSG);
                    bufp += BEEP(bufp);
                    PAINTSCR(screen_buf);
                    message = TRUE;
                  }
                  curbuf[curbuf_read] = '\0';
                  curbuf_read = 0;
                }

        }
        else if (curbuf[curbuf_read] == QUIT)
        {
            temp[0] = '\0';
            return_status = QUIT;
            curbuf[curbuf_read] = '\0';
```

```
            exit_read_function = TRUE;
          }
          else /**   Any other character entered at the keyboard
...   **/
          {
            if (message == FALSE)
            {
              bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
                              INVALID_MSG);
              bufp += GOTOXY(bufp,col ,row);
              message=TRUE;
            }
            curbuf_read++;

          }
      } /**  End of the WHILE loop  **/
      if (OVERFLOW == TRUE && exit_read_function ==
FALSE)
          /** If read enough to fill the size already **/
      {
          if (message == FALSE)
          {
            bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW, EXC_FLD_LIM_MSG);
            PAINTSCR(screen_buf);
            message = TRUE;
          }
          OVERFLOW = FALSE;
          temp[curbuf_consumed] = '\0';
          strcpy(data, temp);
          curbuf_consumed—;
          return_status = curbuf[curbuf_read];
      }
      else
      {
        PAINTSCR(screen_buf);
        bufp = screen_buf;
      }
   }
if (message == TRUE)
 {
       bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW, ERASE_MSG);
       message=FALSE;
       PAINTSCR(screen_buf);
 }
   return (return_status);
}


Display_Results(int txn_type)
{

   void print_neworder_output();
   void print_payment_output();
   void print_ordstat_output();
   void print_delivery_output();
   void print_stocklev_output();

   void (*p_print_function[])() = {
            &print_neworder_output,
            &print_payment_output,
            &print_ordstat_output,
            &print_delivery_output,
            &print_stocklev_output
   };

   (p_print_function[txn_type])();
}

void rearrange_date(char *p){
/*    convert date from  YYYY-MM-DD */
   static char buf[11]="DD-MM-YYYY";
#ifndef DEBUG_LOOPBACK

   buf[0]=p[8];
   buf[1]=p[9];

   buf[3]=p[5];
   buf[4]=p[6];

   buf[6]=p[0];
   buf[7]=p[1];
   buf[8]=p[2];
   buf[9]=p[3];

   strncpy(p,buf,10);
#endif
}
void print_neworder_output()
{

   struct orig_order_line_struct *orig_ol_ptr;
   struct OL_TABLE *ool;
   char *bufp;
   int i,r;
   double ol_amount,total_amount=0.0;
   char temp_bg[2] = " ";


   bufp = output_screen;
/*
   if (oNO->status == ITEM_ERROR)  {
      PAINTSCR(EXECUTION_STATUS_MSG);
      return;
   }
*/
            if (newo_dataptr->header.sql_code != 0) {
                     if (newo_dataptr->header.sql_code == 100)
            PAINTSCR(EXECUTION_STATUS_MSG);
                     else {
            PAINTSCR(ERROR_STATUS_MSG);
                              return;
                     }
            }
/*
   if (newo_dataptr->items_valid != 1)  {
      PAINTSCR(EXECUTION_STATUS_MSG);
      return;
   }
*/
   else {

      bufp += SWITCH_TO_NORMAL(bufp);
      rearrange_date(newo_dataptr->new_date);
      bufp += DISPLAY(bufp,61,2,newo_dataptr->new_date);
      bufp += DISPLAY(bufp,25,3,newo_dataptr->c_last);
      bufp += DISPLAY(bufp,52,3,newo_dataptr->c_credit);
      bufp += DISPLAY_FLOAT(bufp,5,64,3,newo_dataptr-
>c_discount*100.0);
      bufp += DISPLAY_INT(bufp,8,15,4,newo_dataptr->o_id);
      bufp += DISPLAY_INT(bufp,2,42,4,newo_dataptr-
>o_ol_cnt);
```

```
        bufp += DISPLAY_FLOAT(bufp,5,59,4,newo_dataptr-
>w_tax*100.0);
        bufp += DISPLAY_FLOAT(bufp,5,74,4,newo_dataptr-
>d_tax*100.0);

              orig_ol_ptr = orig_ol;
        for (i = 0; i < MAX_OL; i++,orig_ol_ptr++) {
           if (orig_ol_ptr->ol_loc > 0) {

               ool = &newo_dataptr->ol_table[orig_ol_ptr->ol_loc-1];
               r = i + FIRST_OL_ROW;

/*
               ol_amount = orig_ol_ptr->ol_table.ol_quantity * ool-
>i_price;
               total_amount += ol_amount;
*/
                     bufp += DISPLAY(bufp,19,r,ool->name_i);
               bufp += DISPLAY_INT(bufp,3,51,r,ool->s_quantity);
               temp_bg[0] = ool->brand_generic;
               bufp += DISPLAY(bufp,58,r,temp_bg);
               bufp += DISPLAY_MONEY(bufp,6,62,r,ool->price);
               bufp += DISPLAY_MONEY(bufp,7,71,r,ool-
>ol_amount);
            }
/*
            else bufp += BLANK_LINE(bufp,FIRST_OL_ROW + i);
*/
        }
 /*
        total_amount *= oNO->tax_n_discount;

*/
        bufp += DISPLAY_MONEY(bufp,8,70,22,newo_dataptr-
>total);
        bufp += DISPLAY(bufp,20,22,"Success: new-order-
committed");
        *bufp = '\0';
        PAINTSCR(output_screen);
    }

#ifdef DEBUG
    Clog("DBG:  Screen output chars = %d\n",(bufp -
&output_screen[0]));
#endif
}

void print_payment_output()
{

    char *bufp, temp[51], tempbuf2[201];
    char *make_phone(char *), *make_zip(char *);


    bufp = output_screen;

            if (pmt_dataptr->header.sql_code != 0) {
        PAINTSCR(ERROR_STATUS_MSG);
                        return;
            }
            else {

        bufp += SWITCH_TO_NORMAL(bufp);      /* jr */
        rearrange_date(pmt_dataptr->pay_date);

#ifdef DEBUG_LOOPBACK
        bufp += DISPLAY(bufp,7,2,"mm-dd-yy");
        bufp += DISPLAY(bufp,1,5,"STREET 1");
        bufp += DISPLAY(bufp,1,6,"STREET 2");
        bufp += DISPLAY(bufp,1,7,"Menlo Park");
        bufp += DISPLAY(bufp,22,7,"CA ");
        bufp += DISPLAY(bufp,25,7,"94025");
#else
        bufp += DISPLAY(bufp,7,2,pmt_dataptr->pay_date);
        bufp += DISPLAY(bufp,1,5,pmt_dataptr->w_street_1);
        bufp += DISPLAY(bufp,1,6,pmt_dataptr->w_street_2);
        bufp += DISPLAY(bufp,1,7,pmt_dataptr->w_city);
        bufp += DISPLAY(bufp,22,7,pmt_dataptr->w_state);
        bufp += DISPLAY(bufp,25,7,make_zip(pmt_dataptr-
>w_zip));

        bufp += DISPLAY(bufp,42,5,pmt_dataptr->d_street_1);
        bufp += DISPLAY(bufp,42,6,pmt_dataptr->d_street_2);
        bufp += DISPLAY(bufp,42,7,pmt_dataptr->d_city);
        bufp += DISPLAY(bufp,63,7,pmt_dataptr->d_state);
        bufp += DISPLAY(bufp,66,7,make_zip(pmt_dataptr-
>d_zip));

       if (byname == TRUE) bufp +=
DISPLAY_INT(bufp,4,11,9,pmt_dataptr->c_id);
        else bufp += DISPLAY(bufp,29,10,pmt_dataptr->c_last);

        bufp += DISPLAY(bufp,9,10,pmt_dataptr->c_first);
        bufp += DISPLAY(bufp,26,10,pmt_dataptr->c_middle);
        bufp += DISPLAY(bufp,9,11,pmt_dataptr->c_street_1);
        bufp += DISPLAY(bufp,9,12,pmt_dataptr->c_street_2);
        bufp += DISPLAY(bufp,9,13,pmt_dataptr->c_city);
        bufp += DISPLAY(bufp,30,13,pmt_dataptr->c_state);
        bufp += DISPLAY(bufp,33,13,make_zip(pmt_dataptr-
>c_zip));

        pmt_dataptr->c_date[10]='\0';
        rearrange_date(pmt_dataptr->c_date);
        bufp += DISPLAY(bufp,58,10,pmt_dataptr->c_date);
        bufp += DISPLAY(bufp,58,11,pmt_dataptr->c_credit);
        bufp += DISPLAY_FLOAT(bufp,5,58,12,pmt_dataptr-
>c_discount*100.0);
        bufp += DISPLAY(bufp,58,13,make_phone(pmt_dataptr-
>c_phone));
        bufp += DISPLAY_MONEY(bufp,14,55,15,pmt_dataptr-
>c_balance);
        bufp += DISPLAY_MONEY(bufp,13,17,16,pmt_dataptr-
>c_credit_lim);

    /**   Display the first 200 lines if Credit is BC  **/
        if (pmt_dataptr->c_data[0] != NULL)
        {
            bufp += DISPLAY50(bufp,12,18,pmt_dataptr->c_data);
            bufp += DISPLAY50(bufp,12,19,&pmt_dataptr-
>c_data[50]);
            bufp += DISPLAY50(bufp,12,20,&pmt_dataptr-
>c_data[100]);
            bufp += DISPLAY50(bufp,12,21,&pmt_dataptr-
>c_data[150]);
        }
        if (!pmt_dataptr->pay_date) bufp +=
DISPLAY(bufp,MESSAGE_COL,MESSAGE_ROW-
2,BAD_INPUTS);
#endif
        *bufp = '\0';
        PAINTSCR(output_screen);
#ifdef DEBUG
        Clog("DBG:  Screen output chars = %d\n",(bufp -
```

```
&output_screen[0]));
#endif
    }
}

void print_ordstat_output()
{

    struct OL_TABLE2 *sol;
    char *bufp;
    int i=0, r=8;

    bufp = output_screen;

            if (ords_dataptr->header.sql_code != 0) {
        PAINTSCR(ERROR_STATUS_MSG);
                        return;
            }
            else {

        bufp += SWITCH_TO_NORMAL(bufp);    /* jr */

        if (byname == TRUE) bufp +=
DISPLAY_INT(bufp,4,11,3,ords_dataptr->c_id);
        else bufp += DISPLAY(bufp,44,3,ords_dataptr->c_last);

#ifdef DEBUG_LOOPBACK

        bufp += DISPLAY(bufp,24,3,"CUSTOMER NAME");
        bufp += DISPLAY(bufp,41,3,"MIDDLE NAME");
        bufp += DISPLAY_MONEY(bufp,9,15,4,736);
        bufp += DISPLAY_INT(bufp,8,15,6,9);
        bufp += DISPLAY_INT(bufp,4,3,r,103);
        bufp += DISPLAY_INT(bufp,6,14,r,507);
        bufp += DISPLAY_INT(bufp,2,25,r,21);
        bufp += DISPLAY_MONEY(bufp,8,32,r,888);
        bufp += DISPLAY(bufp,47,r,"mm-dd-yy");
#else

        bufp += DISPLAY(bufp,24,3,ords_dataptr->c_first);
        bufp += DISPLAY(bufp,41,3,ords_dataptr->c_middle);
        bufp += DISPLAY_MONEY(bufp,9,15,4,ords_dataptr-
>c_balance);
        bufp += DISPLAY_INT(bufp,8,15,6,ords_dataptr->o_id);

        rearrange_date(ords_dataptr->cur_date);
        bufp += DISPLAY(bufp,38,6,ords_dataptr->cur_date);
        bufp += DISPLAY_INT(bufp,2,76,6,ords_dataptr-
>o_carrier_id);


        for (i=0; i<ords_dataptr->o_ol_cnt; i++)
        {
                    sol = &ords_dataptr->ol_table[i];
                    if (sol->ol_supply_w_id > 0)
                    {

                    bufp += DISPLAY_INT(bufp,4,3,r,sol-
>ol_supply_w_id);
                    bufp += DISPLAY_INT(bufp,6,14,r,sol->ol_i_id);
                    bufp += DISPLAY_INT(bufp,2,25,r,sol-
>ol_quantity);
                    bufp += DISPLAY_MONEY(bufp,8,32,r,sol-
>ol_amount);
                sol->delivery_date[10]='\0';
                rearrange_date(sol->delivery_date);
                        bufp += DISPLAY(bufp,47,r,sol-
>delivery_date);
                        r++;
                    }
        }
#endif
        if (!ords_dataptr->o_ol_cnt) bufp +=
DISPLAY(bufp,MESSAGE_COL,MESSAGE_ROW-
2,BAD_INPUTS);
        *bufp = '\0';
        PAINTSCR(output_screen);
#ifdef DEBUG
        Clog("DBG:  Screen output chars = %d\n",(bufp -
&output_screen[0]));
#endif
    }
}

void print_delivery_output()
{
    char *bufp;
        PAINTSCR(DELIVERY_QUEUED_MSG);
#ifdef DEBUG
    Clog("DBG:  Screen output chars = %d\n",(bufp -
&output_screen[0]));
#endif
}

void print_stocklev_output()
{
    char *bufp;

    bufp = output_screen;

            if (stkl_dataptr->header.sql_code != 0) {
        PAINTSCR(ERROR_STATUS_MSG);
                        return;
            }
            else {

        bufp += SWITCH_TO_NORMAL(bufp);    /* jr */
        bufp += DISPLAY_INT(bufp,3,12,6,stkl_dataptr-
>stock_count);
        *bufp = '\0';
        PAINTSCR(output_screen);
#ifdef DEBUG
        Clog("DBG:  low stock:%d\n", stkl_dataptr->stock_count);
        Clog("DBG:  Screen output chars = %d\n",(bufp -
&output_screen[0]));
#endif
    }
}

char *make_phone(char *data)
{

    static char tempphone[20];

    strncpy(tempphone,data,6);
    tempphone[6] = '-';
    strncpy(&tempphone[7],&data[6],3);
    tempphone[10] = '-';
    strncpy(&tempphone[11],&data[9],3);
    tempphone[14] = '-';
    strncpy(&tempphone[15],&data[12],4);
    tempphone[19] = '\0';

    return tempphone;
}
```

```
char *make_zip(char *data)
{

    static char temp[10];

    strncpy(temp,data,5);
    temp[5] = '-';
     strncpy(&temp[6],&data[5],4);
    temp[10] = '\0';

    return temp;
}
```

screen.e

```
/
*****************************************************************************
******************************** screen.e
*************************************
*****************************************************************************/
/*
**  screen.e — All external definitions for use of screen
functions
*/
extern void    Init_Screen();
extern void    Restore_Screen();
extern void    Paint_Screen(int);
extern void    Send_Menu();
extern int     Get_Menu_Input();
extern int    Get_Form_Data();
extern void    Check_Input_Date();


#define QUIT 3
#define SUBMIT 13
```

screen.h

```
/
*****************************************************************************
******************************** screen.h
*************************************
*****************************************************************************/
/*
**  screen.h — All definitions and structures for screen I/O
*/


#include <sys/termio.h>
extern int tty_in;
extern int tty_out;


#define MAX_FORMS          5

#define MESSAGE_ROW     24
#define MESSAGE_COL      1


#define RTE_SYNCH_CHARACTER '\1'

#define SCRBUF_LEN          1536

#define FIRST_OL_ROW    7


#define BLANK_TO_END_OF_LINE(buf)
sprintf(buf,"\033[K")
#define BLANK_TO_END_OF_SCREEN(buf)
sprintf(buf,"\033[J")
#define CLRSCN(buf)
sprintf(buf,"\033[0m\033[2J")
/*
#define DISPLAY_INT(buf,wid,x,y,ip)
sprintf(buf,"\033[%d\;%dH%d.1d", y, x,ip)
#define DISPLAY_MONEY(buf,wid,x,y,fp)
sprintf(buf,"\033[%d\;%dH$%#wid.2f",y,x,fp)
#define DISPLAY_FLOAT(buf,wid,x,y,fp)
sprintf(buf,"\033[%d\;%dH%#wid.2f", y, x,fp)
*/
#define DISPLAY_INT(buf,wid,x,y,ip)
sprintf(buf,"\033[%d\;%dH%*.1d", y, x,wid,ip)
#define DISPLAY_MONEY(buf,wid,x,y,fp)
sprintf(buf,"\033[%d\;%dH$%#*.2f",y,x,wid,fp)
#define DISPLAY_FLOAT(buf,wid,x,y,fp)
sprintf(buf,"\033[%d\;%dH%#*.2f", y, x,wid,fp)
#define DISPLAY(buf,x,y,txt)
sprintf(buf,"\033[%d\;%dH%s", y, x,txt)
#define DISPLAY50(buf,x,y,txt)
sprintf(buf,"\033[%d\;%dH%50.50s", y, x,txt)
#define DISPLAY_NOW(x,y,txt)                    fprintf(stdout,
"\033[%d\;%dH%s", y, x,txt)
#define PAINTSCR(buf)
write(tty_out,buf,strlen(buf))
#define SWITCH_TO_NORMAL(buf)    sprintf(buf,"\033[0m")
#define SWITCH_TO_REVERSE(buf)   sprintf(buf,"\033[7m")
#define SWITCH_TO_UNDERL(buf)    sprintf(buf,"\033[4m")
#define GOTOXY(buf,x,y)
sprintf(buf,"\033[%d\;%dH", y, x)
#define BEEP(buf)                    sprintf(buf,"\007")
#define BLANK_LINE(buf,line)
sprintf(buf,"\033[%d\;%1H\033[K",line);
#define BLANK_UNDERLINE(buf,x,y,txt)
sprintf(buf,"\033[4m;\033[%d;\%dH%s",y,x,txt);


/**
 Possible status values returned by read functions
 **/

#define CANCELLED      3
#define PREVIOUS_FIELD  4

/**
 Possible key strokes read in by the read functions.  Some are
also returned
     as status from the read functions.
 **/

#define BACKTAB          2     /** Decided to use the CTRL B for
now **/
#define DELETE         127
#define ESCAPE          27
#define LF             10
```

```
#define QUIT            3      /** CNTRL-C Key stroke to quit for
now **/
#define SPACE           32
#define SUBMIT          13     /** Done with screen and submit
key: CR  **/
/*#define SUBMIT                20      /**Using CTRL-T for
debugging purposes  */
#define TAB             9
#define UNDERLINE       95

#define LEAVE_SCREEN_MIN 300    /** Minimum # of
characters to leave screen **/
#define LEAVE_SCREEN_TIMEOUT 2  /** Minimum time to
leave screen, 10=1sec   **/

static int curbuf_consumed = 0;
static int curbuf_read = 0;
static int read_count = 0;
static char curbuf[300];
static BOOLEAN OVERFLOW = FALSE;

static BOOLEAN message ;          /* for suppressing warning
messages */
BOOLEAN payment_input = FALSE;    /* for setting money
condition in read_int */


static struct termio tbufsave;

extern void syserr();

BOOLEAN byname;

void    Init_Screen();
void    Paint_Screen(int);
void    Send_Menu();
int     Get_Menu_Input();



/**
   The following is the struct type used to define the I/O element
   y     is the row position on the screen.
   x     is the column position on the screen.
   len   is the size of the data field in bytes.
   flags is the indicator of manadatory data fields.  '1' means
required.
   dptr  is the pointer to the data.
   fptr  is the pointer to the read funtion for the type of data dptr
         points to.
 **/

typedef struct {
       int     y;
       int     x;
       int     len;
       int     flags;
       int    *dptr;
       int     (*fptr)();
} io_elem;


/*  Temporary structures for storing data that needs manipula-
tion before */
/*    sending off to the TPM                                  */
struct orig_order_line_struct {
    struct io_order_line o_ol;
    short ol_loc;
} orig_ol[MAX_OL];
int int_h_amount;

/*  All the possible messages to print out */
static char MANDATORY_MSG[]=
   "\033[24;1H\033[0mMandatory data field!  Please enter
data.\033[K\033[4m\1";
static char INVALID_MSG[]=
   "\007\033[24;1H\033[0mAn invalid character was entered.
Please enter again.\033[K\033[4m\1";
static char ERASE_MSG[]=
   "\033[24\;1H\033[K\033[4m";
static char MIN1DIGIT_MSG[]=
   "\033[24;1H\033[0mYou must enter atleast 1 digit.  Please
reenter.\033[4m\1";
static char BAD_INPUTS[]=
   "       #### Bad input data was entered — Select again ####
\1\033[2;30H";
static char INCOMPLINE_MSG[]=
   "\033[24;1H\033[0mOrder line is incomplete.  Please
complete the whole line.\033[4m\1";
static char ID_OR_LAST_MSG[]=
   "\033[24;1H\033[0mYou must enter either the Last Name or
the Customer Number.\033[4m\1";
static char EXC_MAX_LFT_DEC_DGT_MSG[]=
   "\033[24;1H\033[0mMaximum digits left of decimal point
already entered.  '.' expected\033[4m\1";
static char EXC_FLD_LIM_MSG[]=
   "\007\033[24\;1H\033[0mMaximum digits already entered.
Tab or <CR> expected\033[4m\1"; /* jr */

static char
EXECUTION_STATUS_MSG[]="\033[0m\033[22\;18HInvalid
Item number. Tx rollbacked";
static char
ERROR_STATUS_MSG[]="\033[0m\033[22\;18H==> Transac-
tion Error <==";
static char
DELIVERY_QUEUED_MSG[]="\033[0m\033[6\;19HDelivery
has been queued";

int read_integer(int, int, int, int, int *);
int read_money(int, int, int, int, float *);
int read_string(int, int, int, int, char *);


char menu_buf[122] = "\033[0m\033[23\;1H1 - New Order   2 -
Payment   3 - Order Status   4 - Delivery   5 - Stock
Level\033[24\;1H9 - Exit\033[24\;35HChoice:\1";

io_elem neworder_inputs[] = {
/*      y       x       len     flags   ptr to data     ptr to read function
*/
/*      -       -       —       ——      —————————       ————————————————
*/
        2,      29,     2,      0,      0,      &read_integer,
        3,      12,     4,      0,      0,      &read_integer,
        7,      3,      4,      0,      0,      &read_integer,
        7,      10,     6,      0,      0,      &read_integer,
        7,      45,     2,      0,      0,      &read_integer,
        8,      3,      4,      0,      0,      &read_integer,
        8,      10,     6,      0,      0,      &read_integer,
        8,      45,     2,      0,      0,      &read_integer,
        9,      3,      4,      0,      0,      &read_integer,
        9,      10,     6,      0,      0,      &read_integer,
        9,      45,     2,      0,      0,      &read_integer,
```

```
        10,     3,     4,     0,     0,    &read_integer,
        10,     10,    6,     0,     0,    &read_integer,
        10,     45,    2,     0,     0,    &read_integer,
        11,     3,     4,     0,     0,    &read_integer,
        11,     10,    6,     0,     0,    &read_integer,
        11,     45,    2,     0,     0,    &read_integer,
        12,     3,     4,     0,     0,    &read_integer,
        12,     10,    6,     0,     0,    &read_integer,
        12,     45,    2,     0,     0,    &read_integer,
        13,     3,     4,     0,     0,    &read_integer,
        13,     10,    6,     0,     0,    &read_integer,
        13,     45,    2,     0,     0,    &read_integer,
        14,     3,     4,     0,     0,    &read_integer,
        14,     10,    6,     0,     0,    &read_integer,
        14,     45,    2,     0,     0,    &read_integer,
        15,     3,     4,     0,     0,    &read_integer,
        15,     10,    6,     0,     0,    &read_integer,
        15,     45,    2,     0,     0,    &read_integer,
        16,     3,     4,     0,     0,    &read_integer,
        16,     10,    6,     0,     0,    &read_integer,
        16,     45,    2,     0,     0,    &read_integer,
        17,     3,     4,     0,     0,    &read_integer,
        17,     10,    6,     0,     0,    &read_integer,
        17,     45,    2,     0,     0,    &read_integer,
        18,     3,     4,     0,     0,    &read_integer,
        18,     10,    6,     0,     0,    &read_integer,
        18,     45,    2,     0,     0,    &read_integer,
        19,     3,     4,     0,     0,    &read_integer,
        19,     10,    6,     0,     0,    &read_integer,
        19,     45,    2,     0,     0,    &read_integer,
        20,     3,     4,     0,     0,    &read_integer,
        20,     10,    6,     0,     0,    &read_integer,
        20,     45,    2,     0,     0,    &read_integer,
        21,     3,     4,     0,     0,    &read_integer,
        21,     10,    6,     0,     0,    &read_integer,
        21,     45,    2,     0,     0,    &read_integer,
        999
};

io_elem payment_inputs[] = {
/*      y       x       len     flags   ptr to data     ptr to read function
*/
/*      -       -       __      ____    _________       ________________
*/
        4,      52,     2,      0,      0,    &read_integer,
              9,              11,             4,              0,              0,
&read_integer,
       10,      29,     16,     0,      0,   &read_string,
        9,      33,     4,      0,      0,  &read_integer,
        9,      54,     2,      0,      0,  &read_integer,
       15,      24,     7,      0,      0,&read_integer,
        999
};
io_elem ordstat_inputs[] = {
/*      y       x       len     flags   ptr to data     ptr to read function
*/
/*      -       -       __      ____    _________       ________________
*/
        2,      29,     2,      0,      0,    &read_integer,
        3,      11,     4,      0,      0,    &read_integer,
        3,      44,     16,     0,      0,   &read_string,
        999
};
io_elem delivery_inputs[] = {
/*      y       x       len     flags   ptr to data     ptr to read function
*/
/*      -       -       __      ____    _________       ________________
*/
        4,      17,     2,      0,      0,    &read_integer,
        999
};
io_elem stocklev_inputs[] = {
/*      y       x       len     flags   ptr to data     ptr to read function
*/
/*      -       -       __      ____    _________       ________________
*/
        4,      24,     2,      0,      0,&read_integer,
         999
};

typedef struct {
        int     x;
        int     y;
        char    *text;
} text_elem;

text_elem NO_text_elem[] = {
                1,              36,             "New Order",
                2,              1,              "Warehouse:",
                2,              19,             "District:",
                2,              55,             "Date:",
                3,              1,              "Customer:",
                3,              19,             "Name:",
                3,              44,             "Credit:",
                3,              57,             "%Disc:",
                4,              1,              "Order Number:",
                4,              25,             "Number of Lines:",
                4,              52,             "W_tax:",
                4,              67,             "D_tax:",
                6,              2,              "Supp_W  Item_Id  Item Name",
                6,              45,             "Qty  Stock  B/G  Price
Amount",
                22,             1,              "Execution Status:",
                22,             62,             "Total:" ,
                0
};

text_elem PT_text_elem[] = {
                1,              38,             "Payment",
                2,              1,              "Date:",
                4,              1,              "Warehouse:",
                4,              42,             "District:",
                9,              1,              "Customer:",
                9,              17,             "Cust-Warehouse:",
                9,              39,             "Cust-District:",
                10,             1,              "Name:",
                10,             50,             "Since:",
                11,             50,             "Credit:",
                12,             50,             "%Disc:",
                13,             50,             "Phone:",
                15,             1,              "Amount Paid:",
                15,             23,             "$",
                15,             37,             "New Cust-Balance:",
                16,             1,              "Credit Limit:",
                18,             1,              "Cust-Data:",
        0
};

text_elem OS_text_elem[] = {
                1,              35,             "Order-Status",
                2,              1,              "Warehouse:",
                2,              19,             "District:",
                3,              1,              "Customer:",
                3,              18,             "Name:",
```

```
        4,      1,      “Cust-Balance:”,
        6,      1,      “Order-Number:”,
        6,      26,     “Entry-Date:”,
        6,      60,     “Carrier_Number:”,
        7,      1,      “Supply-W”,
        7,      14,     “Item-Id”,
        7,      25,     “Qty”,
        7,      33,     “Amount”,
        7,      45,     “Delivery-Date”,
        0
};

text_elem DY_text_elem[] = {
        1,      38,     “Delivery”,
        2,      1,      “Warehouse:”,
        4,      1,      “Carrier Number:”,
        6,      1,      “Execution Status:”,
        0
};

text_elem SL_text_elem[] = {
        1,      38,     “Stock-Level”,
        2,      1,      “Warehouse:”,
        2,      19,     “District:”,
        4,      1,      “Stock Level Threshold:”,
        6,      1,      “low stock:”,
        0
};



struct form_info {
      text_elem *tp;
      char    blank_form[SCRBUF_LEN];
      io_elem *input_elems;
      int     num_input_elems;
};

char output_screen[SCRBUF_LEN];

struct form_info     Forms[MAX_FORMS] = {
   { NO_text_elem,” “, neworder_inputs, 0},
   { PT_text_elem,” “, payment_inputs, 0},
   { OS_text_elem,” “, ordstat_inputs, 0},
   { DY_text_elem,” “, delivery_inputs, 0},
   { SL_text_elem,” “, stocklev_inputs, 0}
};
```

syserr.c

```
/
*******************************************************************************
****************************** syserr.c
************************************
*******************************************************************************/
/*
** syserr.c — Call for exiting program and printing and error
message to
**                the screen
*/

#include <stdio.h>

void syserr(msg)  /* print system call error message and
terminate */
char *msg;
{
   extern int errno, sys_nerr;
   extern char *sys_errlist[];
   extern char tty_name[];

    fprintf(stderr, “\007ERROR: (%s) %s (%d”, tty_name, msg,
errno);
   if (errno > 0 && errno < sys_nerr)
       fprintf(stderr, “;%s)\n”, sys_errlist[errno]);
   else
       fprintf(stderr, “)\n”);
   exit (1);
}
```

tppc.c

```
/
*******************************************************************************
****************************** tpcc.c
************************************
*******************************************************************************/
/*
**  tpcc.c: Main client code for TPC-C.
*/

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include “tpcc.h”
#include “screen.e”
#ifdef TPMONITOR
#include “monitor.e”
#endif
main(int argc, char **argv)
{
  int menu_selection;
  void do_transaction(int);

   initialize(argc,argv);
   Send_Menu();
   while ((menu_selection = Get_Menu_Input()) != 9) {
      if ( (menu_selection < 1) || (menu_selection > 5))
continue;
      do_transaction(menu_selection-1);
      Send_Menu();
   }
   rundown();
}

initialize(int argc, char **argv)
{
 int menu_selection, start,m,n;


#ifdef SOCKET

/* 6/26/95 Changed to only look at SVR4, which starts at /dev/
lat621 */
```

```
   start = atoi(argv[1]) + 620;
    sprintf(tty_name,"/dev/lat/%d",start);


   if ((start-620)%100 == 0) printf("opening device %d %s\n",
start, tty_name);
/*
   printf("opening %s\n", tty_name);
*/
   if ((tty_in=tty_out=open(tty_name, O_RDWR ,0)) == -1)
        syserr("Can't open terminal");

   if (imitate_login_sequence() == -1) argc = 1;
   else argc = 2;
#else
    tty_in = 0;
    tty_out =1;
#endif


    if (argc == 2) {
       user_id = atoi(argv[1]);
       w_id = (atoi(argv[1])-1)/10 + 1;
       d_id = (atoi(argv[1])-1)%10 + 1;
    }

    else  prompt_for_inputs(argc,argv);
/*
#ifndef SOCKET
    else {
        w_id = atoi(argv[1]);
        d_id = atoi(argv[2]);
    }
#endif
*/



#ifdef TPMONITOR

    if (Init_Monitor() ) {
        fprintf(stderr,"Unable to connect to TP Monitor\n\01");
        exit (1);
    }
#endif /*TP MONITOR*/


    Init_Screen();
}

rundown()
{

    Restore_Screen();
#ifdef TPMONITOR

    Rundown_Monitor();
#endif /*TPMONITOR */
}


prompt_for_inputs(int argc,char **argv)
{
int i;
    char *endp;
    static char buffer[30];
    BOOLEAN good_answer = FALSE;

    while (!good_answer) {
       i=write(tty_out,"Enter Usernum: ",15);
       i=read(tty_in,buffer,sizeof(buffer));
       if (i == -1)
        {
           perror(write);
           exit(0);
        }
       user_id = (atoi(buffer));
       w_id = (atoi(buffer)-1)/10 + 1;
       d_id = (atoi(buffer)-1)%10 + 1;
       if (w_id > 0) good_answer = TRUE;
    }
}

#ifdef SOCKET
int imitate_login_sequence()

{
    char buffer[128],temp[128];
    int i;

    if (read(tty_in,buffer,sizeof(buffer)) == -1)
       syserr("read error");

    if (write(tty_out,"login:",6) == -1)
       syserr("write error");
    if (read (tty_in,buffer,sizeof(buffer))== -1)
       syserr("read error");

    if (write(tty_out,"Password:",9) == -1)
       syserr("write error");
    if (read (tty_in,buffer,sizeof(buffer)) == -1)
       syserr("read error");

    if (write(tty_out,">",1) == -1)
       syserr("write error");
    if (read (tty_in,buffer,sizeof(buffer)) == -1)
       syserr("read error");

            /* find out if warehouse & district were specified */
    i = sscanf(buffer,"%s %*s %d %d",temp,&w_id,&d_id);
    if (!strcmp(temp,"csh")) {
       freopen(tty_name,"r",stdin);
       freopen(tty_name,"w",stdout);
       freopen(tty_name,"w",stderr);
       system("exec csh");
       exit (1);
    }
    if (i == 2) i = sscanf(buffer,"%*s %d %d",&w_id,&d_id);

    return i-1;
}

#endif
void do_transaction(int num) {
    int status;

    Paint_Screen(num);
    status=Get_Form_Data(num);
    if (status == QUIT)
       return;

#ifdef TPMONITOR
    status = Snd_Txn_To_Monitor(num);
    if (status == TPM_ERROR) {
```

```
      syserr("\033[24;1H\033[0mTPM Error detected — See
$TPCC_HOME/CLIENTLOG for details\n");
   }

#endif /*TPMONITOR*/
   Display_Results(num);
}
```

tppc.h

```
/
*****************************************************************************
*****************************  tpcc.h
**************************************
*****************************************************************************/
/*
**  tpcc.h:  This header file declares data structures for use in
application
**          and server
*/
#include <sys/types.h>
#include <time.h>



#define BOOLEAN int
#define VMS     0
#define LINEMAX 256

#define FALSE   0
#define TRUE    1

#define NEWORDER    0
#define PAYMENT     1
#define ORDSTAT     2
#define DELIVERY    3
#define STOCKLEV    4

#define MAX_OL  15

/*
**  Terminal IO stuff
*/
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
int tty_in;
int tty_out;



/* error codes */
#define OKAY                    0
#define TPM_ERROR               1
#define SQL_ERROR               2
#define ITEM_ERROR              3


char date_field[80];
char tty_name[11];

int user_id;
int w_id;
int d_id;

int xact_type;

/*
**  Data structures of input data for each transaction type
*/



/*
**  Data structures descriptions for IO data for each transaction
type
**
*/

struct io_order_line {
   int     ol_i_id;
   int     ol_supply_w_id;
   int     ol_quantity;
   char    i_name[25];
   int     s_quantity;
   char    b_g[2];
   double  i_price;
};

struct io_neworder {
   int     w_id;
   int     d_id;
   int     c_id;
   int     o_ol_cnt;
   int     o_all_local;
   struct io_order_line o_ol[MAX_OL];

   char    o_entry_d[20];
   char    c_last[17];
   char    c_credit[3];
   float   c_discount;
   float   w_tax;
   float   d_tax;
   int     o_id;
   int     status;
   double  tax_n_discount;
};

struct  io_payment {
   BOOLEAN byname;
   int     w_id;
   int     d_id;
   int     c_id;
   char    c_last[17];
   int     c_w_id;
   int     c_d_id;
   double  h_amount;

   char    h_date[20];
   char    w_street_1[21];
   char    w_street_2[21];
   char    w_city[21];
   char    w_state[3];
   char    w_zip[10];
   char    d_street_1[21];
   char    d_street_2[21];
   char    d_city[21];
   char    d_state[3];
   char    d_zip[10];
```

```
    char    c_first[17];
    char    c_middle[3];
    char    c_street_1[21];
    char    c_street_2[21];
    char    c_city[21];
    char    c_state[3];
    char    c_zip[10];
    char    c_phone[17];
    char    c_since[11];
    char    c_credit[3];
    double  c_credit_lim;
    float   c_discount;
    double  c_balance;
    char    c_data[201];
};

struct status_order_line {
    int     ol_supply_w_id;
    int     ol_i_id;
    int     ol_quantity;
    double  ol_amount;
    char    ol_delivery_d[11];
};

struct  io_ordstat {
    BOOLEAN byname;
    int     w_id;
    int     d_id;
    int     c_id;
    char    c_last[17];

    char    c_first[17];
    char    c_middle[3];
    double  c_balance;
    int     o_id;
    char    o_entry_d[64];
    int     o_carrier_id;
    int     ol_cnt;
    struct status_order_line    s_ol[MAX_OL];
};

struct  io_delivery  {
    int     w_id;
    int     o_carrier_id;
    int     startq_sec;
    int     startq_usec;
    int     status;
};

struct  io_stocklev {

    int     w_id;
    int     d_id;
    int     threshold;

    int     low_stock;
};

/*
**  Data structure for input & output data
*/

struct io_tpcc {
    int type;
    union {
        struct io_neworder neworder;
        struct io_payment payment;
        struct io_ordstat ordstat;
        struct io_delivery delivery;
        struct io_stocklev stocklev;
    }  info;
};
```

stats.c

```
#ifdef STATS
#include <time.h>
#include <sys/time.h>
#include <stdio.h>
static FILE *out_file, *in_file;

#include <signal.h>
#include <fcntl.h>

#include “stats.h”
volatile static struct timeval init_t;
volatile static         int dont_collect;
volatile static         int do_store;
volatile static         int stat_type;
struct timeval start_t, end_t;
struct timezone start_z, end_z;

static int cnt[TXN_TYPE_CNT][MAX_INT];
static double tot[TXN_TYPE_CNT][MAX_INT],
max[TXN_TYPE_CNT][MAX_INT],
      min[TXN_TYPE_CNT][MAX_INT];

#define CNT         0
#define TOTAL       1
#define MINI        2
#define MAXI        3
#define STAT_CNT 4

char label[][STAT_CNT][9]=
    {“NEWO_CNT”,”NEWO_TOT”,”NEWO_MIN”,”NEWO_MAX”,
        “PAYM_CNT”,”PAYM_TOT”,”PAYM_MIN”,”PAYM_MAX”,
       “ORDS_CNT”,”ORDS_TOT”,”ORDS_MIN”,”ORDS_MAX”,
        “DELV_CNT”,”DELV_TOT”,”DELV_MIN”,”DELV_MAX”,
        “STKL_CNT”,”STKL_TOT”,”STKL_MIN”,”STKL_MAX”};

#ifndef    SAMPLING_RATE
#define    SAMPLING_RATE 30
#endif     /* SAMPLING_RATE */

void start_signal()
{
  struct timeval now_t;
  struct timezone now_z;

#ifdef TRACE
  fprintf(out_file,”Start_signal received\n”);
#endif
  gettimeofday(&now_t, &now_z);
  init_t.tv_sec=now_t.tv_sec;
  init_t.tv_usec=now_t.tv_usec;
#ifdef TRACE
  fprintf(out_file,”init_t: %d, %d\n”, init_t.tv_sec, init_t.tv_usec);
```

```
  fflush(out_file);
#endif
  dont_collect=0;
  do_store=0;
}

/*
 * first time, we get -USR2, we stop collecting
 * second time, we dumps data to file & exit
 */
void end_signal()
{
#ifdef TRACE
  struct timeval now_t;
  struct timezone now_z;

  fprintf(out_file,"End_signal received(%d)\n", do_store);
  gettimeofday(&now_t, &now_z);
  fprintf(out_file,"now_t: %d, %d\n", now_t.tv_sec,
now_t.tv_usec);
  fflush(out_file);
#endif
  if (do_store) {
    store_stats(stat_type);
  }
  else {
    dont_collect=1;
    do_store=1;
    signal(SIGUSR2, end_signal);
  }
}

void init_stats(txn_type)
int txn_type;
{
#ifdef TRACE
  struct timeval now_t;
  struct timezone now_z;
#endif
  char buffer[40];
  int i,j;

  stat_type=txn_type;
  txn_type-=1;        /* two-tier uses 0 for WRONG_INPUT */
  do_store=0;
  dont_collect=1;

  sprintf(buffer,"%s/zserver.%d", getenv("TMPDIR"), getpid());
  out_file=fopen(buffer,"w");

  for(j=0;j<MAX_INT;j++) {
    cnt[txn_type][j]=tot[txn_type][j]=0;
    min[txn_type][j]=999.999;
    max[txn_type][j]=0.0;
  }

  signal(SIGUSR1, start_signal);
  signal(SIGUSR2, end_signal);
#ifdef TRACE
  fprintf(out_file,"signal handlers installed for %d\n", stat_type);
  gettimeofday(&now_t, &now_z);
  fprintf(out_file,"now_t: %d, %d\n", now_t.tv_sec,
now_t.tv_usec);
  fflush(out_file);
#endif
}

void store_stats(txn_type)
int          txn_type;
{
  int j,tot_int;

  txn_type-=1;
  if ( stat_type < 0 )
    fprintf(out_file,"No stats collected.\n");
  else {

    tot_int = MAX_INT;
    while ( (tot_int > 0) && (cnt[txn_type][—tot_int]==0) );
    if (tot_int)
      tot_int++;
#ifdef TRACE
    fprintf(out_file,"Stats collected for %d\n", txn_type);
    fprintf(out_file,"during %d minutes:\n", tot_int);
#endif

    for(j=0;j<tot_int;j++)
      fprintf(out_file,"%s %d\n", label[txn_type][0],
cnt[txn_type][j]);
    for(j=0;j<tot_int;j++)
      fprintf(out_file,"%s %.3f\n", label[txn_type][1],
tot[txn_type][j]);
    for(j=0;j<tot_int;j++)
      fprintf(out_file,"%s %.3f\n", label[txn_type][2],
min[txn_type][j]);
    for(j=0;j<tot_int;j++)
      fprintf(out_file,"%s %.3f\n", label[txn_type][3],
max[txn_type][j]);
  }

  fclose(out_file);
}

void start_stats()
{
  gettimeofday(&start_t,&start_z);
}

static double
getelapsed(s_sec, s_usec, e_sec, e_usec)
int          s_sec, s_usec, e_sec, e_usec;
{
    double s_dbl, e_dbl;

    s_dbl = ((double)s_sec + (s_usec/1000000.0));
    e_dbl = ((double)e_sec + (e_usec/1000000.0));

    if (e_dbl>s_dbl)
      return (e_dbl-s_dbl);
    else
      return (0.00);
}

void update_stats(txn_type)
int          txn_type;
{
  static upd_cnt=0;
  static double init_time, txn_time;
  int idx;

  gettimeofday(&end_t,&end_z);
```

```
  txn_type-=1;
  if (dont_collect)
    return;

  init_time = getelapsed(init_t.tv_sec, init_t.tv_usec,
                    end_t.tv_sec, end_t.tv_usec);

  txn_time = getelapsed(start_t.tv_sec, start_t.tv_usec,
                    end_t.tv_sec, end_t.tv_usec);

  idx = (int)(init_time/SAMPLING_RATE);

  if ( idx>= MAX_INT )
    return;

  cnt[txn_type][idx]++;
  tot[txn_type][idx] += txn_time;

#ifdef TRACE
  if ( (++upd_cnt % 200) == 0 ) {
            fprintf(out_file,"end_t: %d txn_time=%f, upd=%d,
cnt[%d,%d]=%d\n", end_t.tv_sec, txn_time, upd_cnt, txn_type,
idx, cnt[txn_type][idx] );
            fflush(out_file);
  }
#endif
  if (min[txn_type][idx] > txn_time)
    min[txn_type][idx] = txn_time;

  if (max[txn_type][idx] < txn_time)
    max[txn_type][idx] = txn_time;

}

#ifdef MAIN
int rampup_int;
static int svr_cnt[TXN_TYPE_CNT],
svr_max[TXN_TYPE_CNT][MAX_INT],
            svr_min[TXN_TYPE_CNT][MAX_INT];
static char txn_type_label[][15]=
            { "New Order", "Payment", "Order Status", "Delivery",
"Stock Level"};

int get_txn_type(label)
char        *label;
{
  if (strncmp(label,"NEWO",4) == 0)
    return NEW_ORDER;
  else if (strncmp(label,"PAYM",4) == 0)
    return PAYMENT;
  else if (strncmp(label,"ORDS",4) == 0)
    return ORDER_STATUS;
  else if (strncmp(label,"DELV",4) == 0)
    return DELIVERY;
  else if (strncmp(label,"STKL",4) == 0)
    return STOCK_LEVEL;
}

int get_stat_type(label)
char        *label;
{
  if (strncmp(label+5,"CNT",3) == 0)
    return CNT;
  else if (strncmp(label+5,"TOT",3) == 0)
    return TOTAL;
  else if (strncmp(label+5,"MIN",3) == 0)
    return MINI;
  else if (strncmp(label+5,"MAX",3) == 0)
    return MAXI;
}

void read_stats()
{
  char prior_label[20], cur_label[20];
  int idx,int_val;
  int txn_type_idx, stat_type_idx;
  float dbl_val;
  int label_change;
  char buffer[40];

  sprintf(buffer,"%s/zservers", getenv("TMPDIR"));
  in_file = fopen(buffer,"r");

  strcpy(prior_label,"Invalid Label");
  idx=0;

  while (fscanf(in_file, "%s",cur_label) != EOF)
  {
    label_change = FALSE;
    if (strcmp(prior_label,cur_label) != 0)
    {
      idx=0;
      strcpy(prior_label,cur_label);
      label_change = TRUE;
    }

    txn_type_idx = get_txn_type(cur_label)-1;
    stat_type_idx = get_stat_type(cur_label);

    if (stat_type_idx == CNT)
      fscanf(in_file,"%d", &int_val);
    else
      fscanf(in_file,"%f", &dbl_val);

    if (stat_type_idx == CNT)
    {
      if (label_change)
        svr_cnt[txn_type_idx]++;
      cnt[txn_type_idx][idx] += int_val;

      if (svr_min[txn_type_idx][idx] > int_val)
        svr_min[txn_type_idx][idx] = int_val;

      if (svr_max[txn_type_idx][idx] < int_val)
        svr_max[txn_type_idx][idx] = int_val;
    }
    else if (stat_type_idx == TOTAL)
      tot[txn_type_idx][idx] += dbl_val;
    else if (stat_type_idx == MINI)
    {
      if (min[txn_type_idx][idx] > dbl_val)
        min[txn_type_idx][idx] = dbl_val;
    }
    else if (stat_type_idx == MAXI)
    {
      if (max[txn_type_idx][idx] < dbl_val)
        max[txn_type_idx][idx] = dbl_val;
    }
    idx++;
  }
  fclose(in_file);
}

void print_stats()
```

```
{
  int i,j,tot_int;
  float avg, svr_avg;
  int avg_tpm, avg_cnt;
  float avg_rt;

  printf("\nSampling Interval = %d secs\n", SAMPLING_RATE);
  printf("Rampup Intervals = %d\n", rampup_int);

  for (i=0;i<TXN_TYPE_CNT;i++)
  {
    printf("\n%s\t", txn_type_label[i]);
    printf("Server Count: %d\n", svr_cnt[i]);

    printf(
 "\nInterval Count Average Minimum Maximum Svr Avg Svr
Min  Svr Max\n");
    printf(
   "======== ===== ======= ======= ======= =======
=======  =======\n");

    tot_int = MAX_INT;
    while ( (tot_int > 0) && (cnt[i][—tot_int]==0) );
    if (tot_int)
      tot_int++;
    else
      continue;

    avg_cnt=0; avg_rt=0.0; avg_tpm=0;
    for(j=0;j<tot_int;j++)
    {
      avg = tot[i][j] / cnt[i][j];
      svr_avg = (float) cnt[i][j] / svr_cnt[i];
      printf("%8d  %5d  %7.3f  %7.3f  %7.3f  %7.1f  %7d
%7d\n",
        j, cnt[i][j], avg, min[i][j], max[i][j], svr_avg, svr_min[i][j],
        svr_max[i][j]);
      if ( (j>rampup_int) && (j<tot_int-2))
      {
            avg_tpm+=cnt[i][j];
            avg_rt+=avg;
            avg_cnt++;
      }
    }
    printf("AVG[%3d]  %5d  %7.3f\n", avg_cnt, avg_tpm/avg_cnt,
avg_rt/avg_cnt);
  }
  fclose(out_file);
}

main(argv,argc)
int argv;
char *argc[];
{
  int i,j;

  if (argv < 2) {
    printf("*** error arg1: must supply rampup time\n");
    exit(-1);
  }

  rampup_int = (atoi(argc[1]) / SAMPLING_RATE);

  for(i=0;i<TXN_TYPE_CNT;i++) {
    svr_cnt[i]=0;
    for(j=0;j<MAX_INT;j++) {
      cnt[i][j]=tot[i][j]=0;
      min[i][j]=999.999;
      max[i][j]=0.0;
      svr_min[i][j]=9999999;
      svr_max[i][j]=-1 ;
    }
  }


  read_stats();

  print_stats();
}
#endif
#else      /* STATS */
#ifdef MAIN
main()
{
            printf("\nNo stats collected\n");
}
#else
dummy()
{
}
#endif     /* MAIN */
#endif     /* STATS */
```

stats.h

```
#define MAX_INT 400          /* Max # of samples */

#ifndef NEW_ORDER          /* from trans_type.h */
#define NEW_ORDER      1
#define PAYMENT        2
#define ORDER_STATUS   3
#define DELIVERY       4
#define STOCK_LEVEL    5
#endif
#define TXN_TYPE_CNT 5

#ifndef TRUE
#define TRUE       1
#define FALSE      0
#endif

/* forward DECL */
void start_signal();
void end_signal();
void init_stats();
void start_stats();
void update_stats();
void store_stats();
```

tpccfs.c

```
/*
 * Client/Server version of TPC-C
 *
 * This program, the “server front end”,  dequeues requests
from a
 * server work queue and passes them to the “server back
end”,  where the
 * real work gets done.
 *
 */

#include <stdio.h>
#include <unistd.h>
#include <sys/errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <netdb.h>
#include <varargs.h>
#include “do_tpcc.h”
#include “trans_type.h”
#include “utils.h”
#include “ex_trans.h”
#include “databuf.h”
#include “atmi.h”
#ifdef STATS
#include “stats.h”
#endif

#define MAX_NUM_LEN 12

/* Global */
int   sd;   /* socket descriptor for remote host */
int   errno;
char  *User = “ifmx_run”;                          /* User name
on SUT */
char  *Passwd = “ifmx_run”;                        /* Password
on SUT */
int   ServerNumber;                                        /*
Service Number */
char  *ServerProgram;                                      /*
Service Program on SUT */
char  *ServerHostName = “vldb”;            /* SUT Host name */
char  *ClientHostName;                                     /*
Tuxedo machine Host name */

FILE  *LogFp;                                /* Log File pointer */

short int generic_SVC();
/* short int generic_ASYNC_SVC(); */

char msg[128];

static void LOGMSG();
static void OPENLOG();
static void CLOSELOG();

int tpsvrinit(argc, argv)
int    argc;
char **argv;
{

/* The following are all defined as macros */
    struct servent *servent;
    char *rem_cmd       = NULL;
    int  c;
    extern char *optarg;
    int       txn_type;

/* Startup... */
    /*
     * search for the options
     *    “-n” service number
     *    “-S” service program
     *    “-h” host name
              *    “-c” client name
     *    These are specified in “CLOPT” which is specified in
UBBconfig
     */
    while ((c = getopt(argc, argv, “n:c:S:h:”)) != EOF) {
                        switch(c) {
                                case ‘n’:
            ServerNumber = atoi(optarg);
                                break;
            case ‘S’:
                                ServerProgram  = optarg;
                                        break;
                                case ‘h’:
                                ServerHostName = optarg;
                                break;
                                case ‘c’:
                                ClientHostName = optarg;
                                break;
                        }
            }

    OPENLOG(ServerNumber);

    if (ServerNumber < 0)
            {
        LOGMSG(“ERROR: The Server number is not specified (-
n<NUMBER>).”);
                        return 1;
    }
            else
            {
                        /* make sure the order in stats.h is the
                         * same as in Tuxedo conf. file
                         */
                        txn_type = ServerNumber / 100;
#ifdef STATS
                        init_stats(txn_type);
#endif
            }
    if (ClientHostName == NULL)
            {
        LOGMSG(“ERROR: The local Client hostname is not
specified (-c<HOSTNAME>).”);
                        return 1;
    }

    if (ServerProgram == NULL)
            {
        LOGMSG(“ERROR: The remote Server program is not
specified (-S<SERVICE>).”);
                        return 1;
    }
```

```
   if (ServerHostName == NULL)
           {
                   LOGMSG("WARNING:The host is not
specified. Using default '%s'.",

ServerHostName);
   }

   rem_cmd = (char *)malloc(strlen(ServerProgram) +
                                   strlen(ClientHostName)
+ MAX_NUM_LEN);
   if (rem_cmd == NULL) {
                   LOGMSG("ERROR: Ran out of memory
creating remote command.");
                   return 1;
   }

   LOGMSG("ARGS %d %s %s %s", ServerNumber,
ServerProgram,
                                   ServerHostName,
ClientHostName );

   /* create the remote command */
   sprintf(rem_cmd, "%s -n %d -c %s", ServerProgram,
                                   ServerNumber,
ClientHostName);

   /* Fire up the back end... */
   servent = getservbyname("exec", "tcp");

   if ( servent == NULL)
   {
     LOGMSG("ERROR: getservbyname(\"exec\", \"tcp\")
failed.");
     return 1;
   }

   sd = rexec(&ServerHostName, servent->s_port, User,
Passwd, rem_cmd, 0);

   if(sd < 0)
   {
    LOGMSG("ERROR: rexec() call failed.");
    return 1;
   }

   /* Log a message that we have been successful */
   LOGMSG("TPC server has started successfully.");
   return 0;
}

void
NEWO_SVC(svcinfo, txn_type)
TPSVCINFO *svcinfo;
int txn_type;
{
   int returncode;
   returncode = generic_SVC(svcinfo, NEW_ORDER);
           if ( returncode == OS_ERROR )
                   tpreturn(TPFAIL, returncode, NULL, 0, 0);
   else
                   tpreturn(TPSUCCESS, returncode, svcinfo-
>data, svcinfo->len, 0);
}

void
PMT_SVC(svcinfo, txn_type)
TPSVCINFO *svcinfo;
int txn_type;
{
   int returncode;
   returncode = generic_SVC(svcinfo, PAYMENT);
           if ( returncode == OS_ERROR )
                   tpreturn(TPFAIL, returncode, NULL, 0, 0);
   else
                   tpreturn(TPSUCCESS, returncode, svcinfo-
>data, svcinfo->len, 0);
}

void
ORDS_SVC(svcinfo, txn_type)
TPSVCINFO *svcinfo;
int txn_type;
{
   int returncode;
   returncode = generic_SVC(svcinfo, ORDER_STATUS);
           if ( returncode == OS_ERROR )
                   tpreturn(TPFAIL, returncode, NULL, 0, 0);
   else
                   tpreturn(TPSUCCESS, returncode, svcinfo-
>data, svcinfo->len, 0);
}

void
DVRY_SVC(svcinfo, txn_type)
TPSVCINFO *svcinfo;
int txn_type;
{
   int returncode;
   returncode = generic_SVC(svcinfo, DELIVERY);
           if ( returncode == OS_ERROR )
                   tpreturn(TPFAIL, returncode, NULL, 0, 0);
   else
                   tpreturn(TPSUCCESS, returncode, svcinfo-
>data, svcinfo->len, 0);
}

void
STKL_SVC(svcinfo, txn_type)
TPSVCINFO *svcinfo;
int txn_type;
{
   int returncode;
   returncode = generic_SVC(svcinfo, STOCK_LEVEL);
           if ( returncode == OS_ERROR )
                   tpreturn(TPFAIL, returncode, NULL, 0, 0);
   else
                   tpreturn(TPSUCCESS, returncode, svcinfo-
>data, svcinfo->len, 0);
}

short int
generic_SVC(svcinfo, txn_type)
TPSVCINFO *svcinfo;
int txn_type;
{
   int returncode;
   int res;

   /* send to server */
#ifdef DEBUG
   LOGMSG("DEBUG:Sending %d bytes of data to SUT",
svcinfo->len);
#endif
```

```
#ifdef STATS
            start_stats(txn_type);
#endif
    while ( ( (res = write(sd, svcinfo->data, svcinfo->len)) < 0 )
                        && ( errno == EINTR) )
            {
#if 1
                        LOGMSG( "DEBUG: wrote %d bytes\n",
res);
                        dump((char *)(svcinfo->data), res);
#endif
            }

    if (res < svcinfo->len ) {
        LOGMSG("ERROR: generic_SVC: write (%d) to backend
(%d).", res, errno);
                        return OS_ERROR;
    }

    /* Wait for the response */
#ifdef DEBUG
    LOGMSG("DEBUG: Waiting for SUT response");
#endif
    while ( ( (res = read(sd, (void *)svcinfo->data, svcinfo->len))
< 0 )
                        && ( errno == EINTR) );
    if (res < svcinfo->len )
            {
#if 1
                        LOGMSG("ERROR: generic_SVC: read
(%d) from backend (%d)", res, errno);
                        return OS_ERROR;
#endif
    }
#ifdef STATS
    update_stats(txn_type);
#endif
    /* If all went well, respond to client */
    return (((struct data_header *)(svcinfo->data))->returncode);
}

/*****************************************************************
tpsrvdone cleans up after the TPC transaction service
*****************************************************************/
void tpsvrdone()
{
    close(sd);

    /* Log a message saying we are done */
    LOGMSG("TPC server has shutdown successfully.");

    CLOSELOG();
}

static void
LOGMSG( va_alist )
va_dcl
{
    char     buf[128];
    va_list  others;
    char     *fmt;

    va_start(others);

    fmt = va_arg(others, char *);
    vsprintf( buf, fmt, others );
    fprintf( LogFp,"%s\n", buf );
    fflush( LogFp );

    va_end(others);
}

static void
OPENLOG(int id)
{
    char fname[80];

    sprintf( fname,"%s/L%d_%d",getenv( "TUXLOGDIR" ), id/
100, getpid() );
    if ( ( LogFp = fopen( fname,"w" )) == (FILE *)NULL ) {
        perror( "fopen():OPENLOG" );
        exit(99);
    }
    fprintf( LogFp,"%s\n","Opened Log File" );
    fflush( LogFp );

}

static void
CLOSELOG()
{

   fflush( LogFp );
   fclose( LogFp );

}

#if 1
#include <fcntl.h>
dump( a , l)
char *a;
int          l;
{

    int fd;
    char f[20];
    sprintf( f,"%s/x.%06d", getenv("TMPDIR"), getpid() );
    if (( fd = open( f,O_APPEND|O_WRONLY )) < 0 )  {
          printf( "A\n" );
          fd = open( f,O_CREAT|O_WRONLY,0666 );
    }
    write( fd,a,l );
    close( fd );

}
#endif
```

databuf.h

```
#ifndef __TPCC_DATABUF_H__
#define __TPCC_DATABUF_H__

/*
 * BEWARE
 *
 * it's no use using #define for the lenght of the arrays,
 * because that file is also include thru 'SQL INCLUDE' and
 * therefore not pre-processed.
```

```
 */

struct data_header {
        short int       dtype;
        short int       returncode;
        int                             sql_code;
        int                             isam_code;
};

struct OL_TABLE {
        short int       ol_supply_w_id;
        short int       ol_quantity;
        short int       s_quantity;
        int                             ol_i_id;
        char                    name_i[25];
        char                    brand_generic;
        double                  price;
        double                  ol_amount;
};

struct OL_TABLE2 {
        double                  ol_amount;
        int                             ol_i_id;
        short int       ol_supply_w_id;
        short int       ol_quantity;
        char                    delivery_date[20];
};

struct NEWO_DATA {
        struct data_header header;
        short int       w_id;
        short int       d_id;
        int                             c_id;
        short int       o_ol_cnt;
        short int       o_all_local;
        short int       items_valid; /* true if all valid */
        int                             o_id;
        double                  w_tax;
        double                  d_tax;
        double                  total;
        double                  c_discount;
        char                    new_date[20];
        char                    c_last[17];
        char                    c_credit[3];
        struct OL_TABLE ol_table[15];
};

struct PMT_DATA {
        struct data_header header;
        short int       w_id;
        short int       d_id;
        double                  h_amount;
        double                  c_credit_lim;
        double                  c_balance;
        double                  c_discount;
        double                  c_ytd_payment;
        int                             c_id;
        short int       c_w_id;
        short int       c_d_id;
        short int       byname;
        char                    pay_date[20];

        char                    w_name[11];
        char                    w_street_1[21];
        char                    w_street_2[21];
        char                    w_city[21];
        char                    w_state[3];
        char                    w_zip[10];

        char                    d_name[11];
        char                    d_street_1[21];
        char                    d_street_2[21];
        char                    d_city[21];
        char                    d_state[3];
        char                    d_zip[10];

        char                    c_first[17]; /* was
C_LAST_LEN already includes +1 */
        char                    c_middle[3];
        char                    c_last[17];
        char                    c_phone[17];
        char                    c_credit[3];
        char                    c_street_1[21];
        char                    c_street_2[21];
        char                    c_city[21];
        char                    c_state[3];
        char                    c_zip[10];
        short int       c_payment_cnt;
        char                    c_since[20];
        char                    c_data[200];
};

struct ORDS_DATA {
        struct data_header header;
        short int       w_id;
        short int       d_id;
        int                             c_id;
        int                             o_id;
        short int       o_ol_cnt;
        short int       byname;
        short int       o_carrier_id;
        char                    c_last[17];
        char                    c_first[17];
        char                    c_middle[3];
        char                    cur_date[20];
        double                  c_balance;
        struct OL_TABLE2 ol_table[15];
};

struct DVRY_DATA {
        struct data_header header;
        short int       w_id;
        short int       carrier_id;
        int                             startq_sec;
        int                             startq_usec;
};

struct STKL_DATA {
        struct data_header header;
        short int       w_id;
        short int       stkl_d_id;
        short int       threshold;
        int                             stock_count;
};

#endif
```

do_tpcc.h

```
#ifndef __DO_TPCC_H__
#define __DO_TPCC_H__

#define MAX_OL                  15      /* Maximum items in an
order */
#define MIN_OL                  5
#define MAX_FLDS        200     /* Maximum fields in a TPC-C
form */
#define D_PER_W         10

#define ON              1
#define OFF             0

#define TRUE            1
#define FALSE           0

#define YES             1
#define NO              0

#define INSIZE  1024

#define RETRY_MSG       "Try again..."
#define COMMIT_NEWO  99
#define COMMIT_PAYMENT  88
#define COMMIT_DELIVERY  77
#define COMMIT_ORDERSTAT  66
#define COMMIT_STOCKLEV  55

#define INVALID_NEWO  1
#define SQL_ERROR    -1
#define OS_ERROR     -2
#define CANCEL       -1

#endif
```

ex_trans.ec

```
/
*=================================================================*
|   ex_trans.ec - This module includes procedures executing
TPC-C                           |
|                                                 transactions.  It
contains SQL statements, therefore, the  |
|                                                 implementation is
Database specific.
|
|

                                                        |
|               "ex_trans.ec" contains the following procedures:
                                                |
|

                                                        |
|                       source code will "look good" if you :set
tabstop=4                                                       |
*=================================================================*/

#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <time.h>
#include <sys/time.h>
#include <datetime.h>
#include <signal.h>
#include <errno.h>
#include <sys/ipc.h>
#include "trans_type.h"
#include "ex_trans.h"
#include "do_tpcc.h"

#define INPUT  0
#define OUTPUT 1
#define MAX_STR_LEN         128

static int  dvry_log;
static int  status_log;
static void OpenDeliveryLog();
static void CloseDeliveryLog();
static void server_loop();


EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE databuf.h;
/* variable set only once in the beginning of the application.  */
dtime_t    infx_time;
char s[1024];

EXEC SQL END DECLARE SECTION;

#define MAX_DATA_LEN sizeof(struct NEWO_DATA)
#define SQL_CHECK(line)
                                                \
            if (SQCODE != 0) {
                                                \
                        LOGMSG2("*** ERROR at line %d\n", line);
            \
                        goto sqlerr;
                                                        \
            }


static      int                     ESQL_prep_dvry();
static      int                     ESQL_prep_newo();
static      int                     ESQL_prep_pmt();
static      int                     ESQL_prep_ords();
static      int                     ESQL_prep_stkl();

static      int                     ESQL_init(int sid);
static      void        ESQL_close(int sid);

static      char        msg[100];

/
*****************************************************************************/
/*
UTILITY FUNCTIONS
                                        */
/
*****************************************************************************/
```

```
/
*****************************************************************************
 * log handling functions / macros .
 *****************************************************************************/
static      FILE *LogFp;
#define    LOGMSG( msg )                { fprintf( LogFp, "%s",
msg); fflush( LogFp ); }
#define    LOGMSG2( fmt, val)           { fprintf( LogFp, fmt ,
val); fflush( LogFp ); }
static void
OPENLOG(int id, char *clt)
{
            char fname[80];
            char *envptr;

            if ( (envptr = getenv("SRVLOGDIR")) == NULL ) {
                        perror( "getenv():SRVLOGDIR" );
                        exit(99);
            }
            sprintf( fname, "%s/L%1d_%d.%s", envptr, id/100,
getpid(), clt );
            if ( ( LogFp = fopen( fname, "w" )) == (FILE *)NULL ) {
                        perror( "fopen():OPENLOG" );
                        exit(99);
            }
            LOGMSG( "Opened Log File\n" )
}

static void
CLOSELOG()
{
            fflush( LogFp );
            fclose( LogFp );
}

#ifdef XDUMP
/
*****************************************************************************
 * HexDump - dumps out a buffer page.
 *****************************************************************************/
static      void
HexDump( pagebuffer, l )
char pagebuffer[];
int l;
{

            int p, i, j;
            char locbuf[16];
            char linbuf[80];

            fprintf( LogFp,
            "\n\n<——— HEXADECIMAL ———————————————
———> <— CHARACTER ->\n\n"
                                    );

            memset( linbuf, 0x0, 80 );
            p = j = 0;
            for( i=0;i<=l;i++ ) {

                        if (  i && !( i % 16 ) ) {
                           for( p=j;p<i;p++ ) {
                                    sprintf( locbuf, "%c", (
!isprint(pagebuffer[p]) ?
                                                                        '.' :
pagebuffer[p] ) );
                                    strcat( linbuf, locbuf );
                           }
                           fprintf( LogFp, "%s\n", linbuf );
                           memset( linbuf, 0x0, 80 );
                           j = i;

                        }
                        sprintf( locbuf, "%02x ", (unsigned
char)pagebuffer[i] );
                        strcat( linbuf, locbuf );
            }
}
#endif

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

static      int data_len            = 0;  /* size of each record */
static      int (*service_routine)();  /* function to service request
*/

#define    SQCODE                       (sqlca.sqlcode)
#define    ISCODE                       (sqlca.sqlerrd[1])
#define    DEADLOCK                             -143
#define    TIMEOUT                                      -
346
#define    DEADLOCK_TIMEOUT        -154
#define    MAX_TRY                                      3
#define    DO_ROLLBACK                          1
#define    DONT_ROLLBACK           0
#define    NO_OUTSTANDING                       0
#define    SOMETHING_WRONG                      -1
#define    MAX_DVRY_LOG_MSG        250

static      int delivery_trans_cnt = 0;

/
*****************************************************************************/
/*
SQL HANDLING FUNCTIONS
                                    */
/
*****************************************************************************/

/* Connect and Prepare */
static      int
ESQL_init(sid)
int sid;
{

            int retval;

            LOGMSG( "Attempting Connect To Informix
Database\n" )

            EXEC SQL DATABASE tpcc;
            if (sqlca.sqlcode)
            {
                        retval=1;
                        return retval;
            }

            LOGMSG( "Connected To Informix Database\n" )

            EXEC SQL SET ISOLATION TO REPEATABLE
READ;
```

```
	EXEC SQL SET LOCK MODE TO WAIT;

	/*
	 * Prepare SQL statements.
	 */

	LOGMSG( “Attempting Prepare Stmt\n” )

	switch(sid) {
		case NEW_ORDER:
			retval = ESQL_prep_newo();
		break;
		case PAYMENT:
			retval = ESQL_prep_pmt();
		break;
		case ORDER_STATUS:
			retval = ESQL_prep_ords();
		break;
		case DELIVERY:
			OpenDeliveryLog();
			retval = ESQL_prep_dvry();
		break;
		case STOCK_LEVEL:
			retval = ESQL_prep_stkl();
		break;
	}

	if ( retval )
		return retval;

	LOGMSG( “Stmt Prepared\n” )

	/*
	 * set the qualifier of the datetime variable which will
be used in
	 * the application.
	 */
	infx_time.dt_qual = TU_DTENCODE(TU_YEAR,
TU_SECOND);

	return 0;
}

/* Disconnect */

static	void
ESQL_close(sid)
int sid;
{

	LOGMSG( “Disconnecting From Informix Database
tpcc\n” )
	EXEC SQL CLOSE DATABASE;
	if ( sid == DELIVERY )
	  CloseDeliveryLog();

}

/* Prepare Database Stuff */

static	int
ESQL_prep_newo()
{

	LOGMSG( “Preparing NewOrder\n” )
	EXEC SQL WHENEVER SQLERROR GOTO sqlerr;

	sprintf(s, “EXECUTE PROCEDURE neworder
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?);”);

	EXEC SQL SET OPTIMIZATION HIGH;
	EXEC SQL PREPARE sneworder FROM :s;
	EXEC SQL SET OPTIMIZATION LOW;
	EXEC SQL SET DATALOCKCACHE 300;
	EXEC SQL COMMIT WORK;
	EXEC SQL WHENEVER SQLERROR continue;
	LOGMSG( “Preparing NewOrder Successful\n” )
	return 0;

sqlerr:

	sprintf( msg,”PREPARE FAIL NEWORDER SQL %d
ISAM %d\n”,SQCODE,ISCODE );
	LOGMSG( msg )
	return 1;
}


static	int
ESQL_prep_pmt()
{

	LOGMSG( “Preparing Payment\n” )
	EXEC SQL WHENEVER SQLERROR GOTO sqlerr;

	sprintf(s, “EXECUTE PROCEDURE pay-
ment(?,?,?,?,?,?,?,?,?,?);”);

	EXEC SQL SET OPTIMIZATION HIGH;
	EXEC SQL PREPARE spayment FROM :s;
	EXEC SQL SET OPTIMIZATION LOW;
	EXEC SQL SET DATALOCKCACHE 400;
	EXEC SQL COMMIT WORK;
	EXEC SQL WHENEVER SQLERROR continue;
	LOGMSG( “Preparing Payment Successful\n” )
	return 0;

sqlerr:

	sprintf( msg,”PREPARE FAIL PAYMENT SQL %d
ISAM %d\n”,SQCODE,ISCODE );
	LOGMSG( msg )
	return 1;

}


static	int
ESQL_prep_ords()
{

	LOGMSG( “Preparing OrderStatus\n” )
	EXEC SQL WHENEVER SQLERROR GOTO sqlerr;

	sprintf(s, “EXECUTE PROCEDURE
order_status(?,?,?,?,?);”);

	EXEC SQL SET OPTIMIZATION HIGH;
	EXEC SQL PREPARE sordstat FROM :s;
	EXEC SQL SET OPTIMIZATION LOW;
	EXEC SQL SET DATALOCKCACHE 200;
	EXEC SQL COMMIT WORK;

	EXEC SQL WHENEVER SQLERROR continue;
```

```
        LOGMSG( “Preparing OrderStatus Successful\n” )
        return (0);

sqlerr:

        sprintf( msg,”PREPARE FAIL ORDERSTATUS SQL
%d ISAM %d\n”,SQCODE,ISCODE );
        LOGMSG( msg )
        return 1;
}


static  int
ESQL_prep_stkl()
{
        LOGMSG( “Preparing StockLevel\n” )
        EXEC SQL WHENEVER SQLERROR GOTO sqlerr;

        sprintf(s, “EXECUTE PROCEDURE
stock_level(?,?,?);”);

        EXEC SQL SET OPTIMIZATION HIGH;

        /* use Version 4 Stored Procedures */

#if 1
        EXEC SQL CREATE TEMP TABLE stk_lvl_tmp (
t_i_id INT ) WITH NO LOG;
#endif

        EXEC SQL PREPARE sstocklev FROM :s;
        EXEC SQL SET OPTIMIZATION LOW;
        EXEC SQL SET DATALOCKCACHE 801;
        EXEC SQL SET ISOLATION TO COMMITTED
READ;
        EXEC SQL COMMIT WORK;
        EXEC SQL WHENEVER SQLERROR continue;
        LOGMSG( “Preparing StockLevel Successful\n” )
        return (0);

sqlerr:

        sprintf( msg,”PREPARE FAIL STOCKLEVEL SQL
%d ISAM %d\n”,SQCODE,ISCODE );
        LOGMSG( msg )
        return 1;

}

#ifdef MULTITRIP
static int
ESQL_prep_dvry()
{
   LOGMSG( “Preparing Delivery\n” );
   EXEC SQL WHENEVER SQLERROR GOTO sqlerr;

   sprintf(s, “EXECUTE PROCEDURE delivery (?,?,?,?);”);

   EXEC SQL SET OPTIMIZATION HIGH;
   EXEC SQL PREPARE sxdelivery FROM :s;
   EXEC SQL SET OPTIMIZATION LOW;
   EXEC SQL SET DATALOCKCACHE 200;
   EXEC SQL COMMIT WORK;
   EXEC SQL WHENEVER SQLERROR continue;
   LOGMSG( “Preparing MultiTrip Delivery Successful\n” );
   return 0;

sqlerr:

   sprintf( msg,”PREPARE FAIL MULTITRIP DELIVERY SQL
%d ISAM %d\n”,SQCODE,ISCODE );
   LOGMSG( msg );
   return 1;

}
#else
static int
ESQL_prep_dvry()
{

   LOGMSG( “Preparing Delivery\n” );
   EXEC SQL WHENEVER SQLERROR GOTO sqlerr;

   sprintf(s, “EXECUTE PROCEDURE delivery (?,?,?);”);

   EXEC SQL SET OPTIMIZATION HIGH;
   EXEC SQL PREPARE sdelivery FROM :s;
   EXEC SQL SET OPTIMIZATION LOW;
   EXEC SQL SET DATALOCKCACHE 250;
   EXEC SQL COMMIT WORK;
   EXEC SQL WHENEVER SQLERROR continue;
   LOGMSG( “Preparing Delivery Successful\n” );
   return 0;

sqlerr:

   sprintf( msg,”PREPARE FAIL DELIVERY SQL %d ISAM
%d\n”,SQCODE,ISCODE );
   LOGMSG( msg );
   return 1;

}
#endif

/* New Order Transaction */

static  int
ESQL_NEWO_SVC(newo_ptr)
$parameter register struct NEWO_DATA *newo_ptr;
{

        EXEC SQL BEGIN DECLARE SECTION;
                int             sql_code;
                int             isam_code;
        EXEC SQL END DECLARE SECTION;

        register int j;
        int             retry;

        retry = 0;

deadlock_retry:

        j = 0;

        for (j = newo_ptr->o_ol_cnt; j < MAX_OL; j++)
                newo_ptr->ol_table[j].ol_i_id = -1;

        dtcurrent(&infx_time);
        dttoasc(&infx_time, newo_ptr->new_date);

        EXEC SQL EXECUTE sneworder INTO
        :sql_code,
```

```
            :isam_code,
            :newo_ptr->o_id,
            :newo_ptr->w_tax,
            :newo_ptr->d_tax,
            :newo_ptr->c_discount,
            :newo_ptr->c_last,
            :newo_ptr->c_credit,
            :newo_ptr->total,
            :newo_ptr->ol_table[0].ol_amount, :newo_ptr-
>ol_table[0].price,
            :newo_ptr->ol_table[0].s_quantity, :newo_ptr-
>ol_table[0].name_i,
            :newo_ptr->ol_table[0].brand_generic,

            :newo_ptr->ol_table[1].ol_amount, :newo_ptr-
>ol_table[1].price,
            :newo_ptr->ol_table[1].s_quantity, :newo_ptr-
>ol_table[1].name_i,
            :newo_ptr->ol_table[1].brand_generic,

            :newo_ptr->ol_table[2].ol_amount, :newo_ptr-
>ol_table[2].price,
            :newo_ptr->ol_table[2].s_quantity, :newo_ptr-
>ol_table[2].name_i,
            :newo_ptr->ol_table[2].brand_generic,

            :newo_ptr->ol_table[3].ol_amount, :newo_ptr-
>ol_table[3].price,
            :newo_ptr->ol_table[3].s_quantity, :newo_ptr-
>ol_table[3].name_i,
            :newo_ptr->ol_table[3].brand_generic,

            :newo_ptr->ol_table[4].ol_amount, :newo_ptr-
>ol_table[4].price,
            :newo_ptr->ol_table[4].s_quantity, :newo_ptr-
>ol_table[4].name_i,
            :newo_ptr->ol_table[4].brand_generic,

            :newo_ptr->ol_table[5].ol_amount, :newo_ptr-
>ol_table[5].price,
            :newo_ptr->ol_table[5].s_quantity, :newo_ptr-
>ol_table[5].name_i,
            :newo_ptr->ol_table[5].brand_generic,

            :newo_ptr->ol_table[6].ol_amount, :newo_ptr-
>ol_table[6].price,
            :newo_ptr->ol_table[6].s_quantity,:newo_ptr-
>ol_table[6].name_i,
            :newo_ptr->ol_table[6].brand_generic,

            :newo_ptr->ol_table[7].ol_amount, :newo_ptr-
>ol_table[7].price,
            :newo_ptr->ol_table[7].s_quantity,:newo_ptr-
>ol_table[7].name_i,
            :newo_ptr->ol_table[7].brand_generic,

            :newo_ptr->ol_table[8].ol_amount, :newo_ptr-
>ol_table[8].price,
            :newo_ptr->ol_table[8].s_quantity,:newo_ptr-
>ol_table[8].name_i,
            :newo_ptr->ol_table[8].brand_generic,

            :newo_ptr->ol_table[9].ol_amount, :newo_ptr-
>ol_table[9].price,
            :newo_ptr->ol_table[9].s_quantity,:newo_ptr-
>ol_table[9].name_i,
            :newo_ptr->ol_table[9].brand_generic,

            :newo_ptr->ol_table[10].ol_amount, :newo_ptr-
>ol_table[10].price,
            :newo_ptr->ol_table[10].s_quantity, :newo_ptr-
>ol_table[10].name_i,
            :newo_ptr->ol_table[10].brand_generic,

            :newo_ptr->ol_table[11].ol_amount, :newo_ptr-
>ol_table[11].price,
            :newo_ptr->ol_table[11].s_quantity, :newo_ptr-
>ol_table[11].name_i,
            :newo_ptr->ol_table[11].brand_generic,

            :newo_ptr->ol_table[12].ol_amount, :newo_ptr-
>ol_table[12].price,
            :newo_ptr->ol_table[12].s_quantity,:newo_ptr-
>ol_table[12].name_i,
            :newo_ptr->ol_table[12].brand_generic,

            :newo_ptr->ol_table[13].ol_amount, :newo_ptr-
>ol_table[13].price,
            :newo_ptr->ol_table[13].s_quantity,:newo_ptr-
>ol_table[13].name_i,
            :newo_ptr->ol_table[13].brand_generic,

            :newo_ptr->ol_table[14].ol_amount, :newo_ptr-
>ol_table[14].price,
            :newo_ptr->ol_table[14].s_quantity,:newo_ptr-
>ol_table[14].name_i,
            :newo_ptr->ol_table[14].brand_generic

            USING

            :newo_ptr->o_ol_cnt,
            :newo_ptr->ol_table[0].ol_i_id, :newo_ptr-
>ol_table[0].ol_supply_w_id,
            :newo_ptr->ol_table[0].ol_quantity,
            :newo_ptr->ol_table[1].ol_i_id, :newo_ptr-
>ol_table[1].ol_supply_w_id,
            :newo_ptr->ol_table[1].ol_quantity,
            :newo_ptr->ol_table[2].ol_i_id, :newo_ptr-
>ol_table[2].ol_supply_w_id,
            :newo_ptr->ol_table[2].ol_quantity,
            :newo_ptr->ol_table[3].ol_i_id, :newo_ptr-
>ol_table[3].ol_supply_w_id,
            :newo_ptr->ol_table[3].ol_quantity,
            :newo_ptr->ol_table[4].ol_i_id, :newo_ptr-
>ol_table[4].ol_supply_w_id,
            :newo_ptr->ol_table[4].ol_quantity,
            :newo_ptr->ol_table[5].ol_i_id, :newo_ptr-
>ol_table[5].ol_supply_w_id,
            :newo_ptr->ol_table[5].ol_quantity,
            :newo_ptr->ol_table[6].ol_i_id, :newo_ptr-
>ol_table[6].ol_supply_w_id,
            :newo_ptr->ol_table[6].ol_quantity,
            :newo_ptr->ol_table[7].ol_i_id, :newo_ptr-
>ol_table[7].ol_supply_w_id,
            :newo_ptr->ol_table[7].ol_quantity,
            :newo_ptr->ol_table[8].ol_i_id, :newo_ptr-
>ol_table[8].ol_supply_w_id,
            :newo_ptr->ol_table[8].ol_quantity,
            :newo_ptr->ol_table[9].ol_i_id,:newo_ptr-
>ol_table[9].ol_supply_w_id,
            :newo_ptr->ol_table[9].ol_quantity,
            :newo_ptr->ol_table[10].ol_i_id, :newo_ptr-
>ol_table[10].ol_supply_w_id,
            :newo_ptr->ol_table[10].ol_quantity,
```

```
        :newo_ptr->ol_table[11].ol_i_id, :newo_ptr-
>ol_table[11].ol_supply_w_id,
        :newo_ptr->ol_table[11].ol_quantity,
        :newo_ptr->ol_table[12].ol_i_id, :newo_ptr-
>ol_table[12].ol_supply_w_id,
        :newo_ptr->ol_table[12].ol_quantity,
        :newo_ptr->ol_table[13].ol_i_id, :newo_ptr-
>ol_table[13].ol_supply_w_id,
        :newo_ptr->ol_table[13].ol_quantity,
        :newo_ptr->ol_table[14].ol_i_id, :newo_ptr-
>ol_table[14].ol_supply_w_id,
        :newo_ptr->ol_table[14].ol_quantity,
        :newo_ptr->w_id,
        :newo_ptr->d_id,
        :newo_ptr->c_id,
        :infx_time;

        newo_ptr->header.sql_code = sql_code;
        newo_ptr->header.isam_code = isam_code;

        if (sql_code == 0)
        {
                newo_ptr->items_valid = 1;
                newo_ptr->header.returncode =
COMMIT_NEWO;
                return 0;

        }

        if ( (isam_code == DEADLOCK) || (isam_code ==
DEADLOCK_TIMEOUT ))
        {

                sleep(1);
                retry++;
                sprintf( msg,"New Order Transaction Retry
%d",retry );
                LOGMSG( msg )
                sprintf(msg, "\tw_id= %d - d_id = %d - c_id
= %d\n",
                        newo_ptr->w_id, newo_ptr->d_id,
newo_ptr->c_id );
                LOGMSG( msg )
                goto deadlock_retry;

        }
        else {

                newo_ptr->items_valid = 0;
                newo_ptr->header.returncode =
SQL_ERROR;
                sprintf(msg, "NEW ORDER Transaction
Fail SQL %d ISAM %d",SQCODE,ISCODE);
                LOGMSG( msg )
                sprintf(msg, "\tw_id= %d - d_id = %d - c_id
= %d - %s",
                        newo_ptr->w_id, newo_ptr->d_id,
newo_ptr->c_id, newo_ptr->new_date );
                LOGMSG( msg )
                sprintf(msg, "\tProcedure return SQL Code:
%d ISAM Code %d\n",

sql_code, isam_code);
                LOGMSG( msg )
                return SQL_ERROR;

        }
}

/* Payment Transaction */

static  int
ESQL_PMT_SVC(pmt_ptr)
$parameter register struct PMT_DATA *pmt_ptr;
{

        EXEC SQL BEGIN DECLARE SECTION;
          int       sql_code, isam_code;
        EXEC SQL END DECLARE SECTION;
        int                     retry;

        retry = 0;

deadlock_retry:

        dtcurrent( &infx_time );
        dttoasc(&infx_time, pmt_ptr->pay_date);

        EXEC SQL EXECUTE spayment INTO
                :sql_code,
                :isam_code,
                :pmt_ptr->c_id,
                :pmt_ptr->c_last,
                :pmt_ptr->w_street_1,
                :pmt_ptr->w_street_2,
                :pmt_ptr->w_city,
                :pmt_ptr->w_state,
                :pmt_ptr->w_zip,
                :pmt_ptr->d_street_1,
                :pmt_ptr->d_street_2,
                :pmt_ptr->d_city,
                :pmt_ptr->d_state,
                :pmt_ptr->d_zip,
                :pmt_ptr->c_first,
                :pmt_ptr->c_middle,
                :pmt_ptr->c_street_1,
                :pmt_ptr->c_street_2,
                :pmt_ptr->c_city,
                :pmt_ptr->c_state,
                :pmt_ptr->c_zip,
                :pmt_ptr->c_phone,
                :pmt_ptr->c_since,
                :pmt_ptr->c_credit,
                :pmt_ptr->c_credit_lim,
                :pmt_ptr->c_discount,
                :pmt_ptr->c_balance,
                :pmt_ptr->c_data
         USING
                :pmt_ptr->d_id,
                :pmt_ptr->c_id,
                :pmt_ptr->c_last,
                :pmt_ptr->c_d_id,
                :pmt_ptr->c_w_id,
                :pmt_ptr->h_amount,
                :pmt_ptr->w_id,
                :pmt_ptr->byname,
                :infx_time;

        pmt_ptr->header.sql_code   = sql_code;
        pmt_ptr->header.isam_code  = isam_code;

        if (sql_code == 0) {
                pmt_ptr->header.returncode =
```

```
COMMIT_PAYMENT;
                return 0;
        }

        if ( (isam_code == DEADLOCK) || (isam_code ==
DEADLOCK_TIMEOUT )) {

                sleep(1);
                retry++;
                sprintf( msg,"Payment Transaction Retry
%d\n",retry );
                LOGMSG( msg )
                goto deadlock_retry;

        }
        else
        {
                pmt_ptr->header.returncode =
SQL_ERROR;
                sprintf(msg,"PAYMENT Transaction Fail
SQL %d ISAM %d",SQCODE,ISCODE);
                LOGMSG( msg )
                sprintf(msg, "\td_id= %d - c_id = %d -
c_last = %s",
                        pmt_ptr->d_id, pmt_ptr->c_id,
pmt_ptr->c_last );
                LOGMSG( msg )
                sprintf(msg, "\tProcedure return SQL Code:
%d ISAM Code %d\n",
                        sql_code, isam_code);
                LOGMSG( msg )

                return SQL_ERROR;
        }

}

/* Order Status Transactiom */

static  int
ESQL_ORDS_SVC(ord_ptr)
$parameter register struct ORDS_DATA *ord_ptr;
{

        EXEC SQL BEGIN DECLARE SECTION;
                int
 sql_code,isam_code;
        EXEC SQL END DECLARE SECTION;
        int                             i;
        int                             retry;

        retry = 0;

deadlock_retry:

        EXEC SQL EXECUTE sordstat INTO
                 :sql_code,
                 :isam_code,
                 :ord_ptr->c_id,
                 :ord_ptr->c_first,
                 :ord_ptr->c_middle,
                 :ord_ptr->c_last,
                 :ord_ptr->c_balance,
                 :ord_ptr->o_id,
                 :ord_ptr->cur_date,
                 :ord_ptr->o_carrier_id,
                 :ord_ptr->o_ol_cnt,
                 :ord_ptr->ol_table[0].ol_i_id,
                 :ord_ptr->ol_table[0].ol_supply_w_id,
                 :ord_ptr->ol_table[0].ol_quantity,
                 :ord_ptr->ol_table[0].ol_amount,
                 :ord_ptr->ol_table[0].delivery_date,

                 :ord_ptr->ol_table[1].ol_i_id,
                 :ord_ptr->ol_table[1].ol_supply_w_id,
                 :ord_ptr->ol_table[1].ol_quantity,
                 :ord_ptr->ol_table[1].ol_amount,
                 :ord_ptr->ol_table[1].delivery_date,

                 :ord_ptr->ol_table[2].ol_i_id,
                 :ord_ptr->ol_table[2].ol_supply_w_id,
                 :ord_ptr->ol_table[2].ol_quantity,
                 :ord_ptr->ol_table[2].ol_amount,
                 :ord_ptr->ol_table[2].delivery_date,

                 :ord_ptr->ol_table[3].ol_i_id,
                 :ord_ptr->ol_table[3].ol_supply_w_id,
                 :ord_ptr->ol_table[3].ol_quantity,
                 :ord_ptr->ol_table[3].ol_amount,
                 :ord_ptr->ol_table[3].delivery_date,

                 :ord_ptr->ol_table[4].ol_i_id,
                 :ord_ptr->ol_table[4].ol_supply_w_id,
                 :ord_ptr->ol_table[4].ol_quantity,
                 :ord_ptr->ol_table[4].ol_amount,
                 :ord_ptr->ol_table[4].delivery_date,

                 :ord_ptr->ol_table[5].ol_i_id,
                 :ord_ptr->ol_table[5].ol_supply_w_id,
                 :ord_ptr->ol_table[5].ol_quantity,
                 :ord_ptr->ol_table[5].ol_amount,
                 :ord_ptr->ol_table[5].delivery_date,

                 :ord_ptr->ol_table[6].ol_i_id,
                 :ord_ptr->ol_table[6].ol_supply_w_id,
                 :ord_ptr->ol_table[6].ol_quantity,
                 :ord_ptr->ol_table[6].ol_amount,
                 :ord_ptr->ol_table[6].delivery_date,

                 :ord_ptr->ol_table[7].ol_i_id,
                 :ord_ptr->ol_table[7].ol_supply_w_id,
                 :ord_ptr->ol_table[7].ol_quantity,
                 :ord_ptr->ol_table[7].ol_amount,
                 :ord_ptr->ol_table[7].delivery_date,

                 :ord_ptr->ol_table[8].ol_i_id,
                 :ord_ptr->ol_table[8].ol_supply_w_id,
                 :ord_ptr->ol_table[8].ol_quantity,
                 :ord_ptr->ol_table[8].ol_amount,
                 :ord_ptr->ol_table[8].delivery_date,

                 :ord_ptr->ol_table[9].ol_i_id,
                 :ord_ptr->ol_table[9].ol_supply_w_id,
                 :ord_ptr->ol_table[9].ol_quantity,
                 :ord_ptr->ol_table[9].ol_amount,
                 :ord_ptr->ol_table[9].delivery_date,

                 :ord_ptr->ol_table[10].ol_i_id,
                 :ord_ptr->ol_table[10].ol_supply_w_id,
                 :ord_ptr->ol_table[10].ol_quantity,
                 :ord_ptr->ol_table[10].ol_amount,
                 :ord_ptr->ol_table[10].delivery_date,
```

```
                    :ord_ptr->ol_table[11].ol_i_id,
                    :ord_ptr->ol_table[11].ol_supply_w_id,
                    :ord_ptr->ol_table[11].ol_quantity,
                    :ord_ptr->ol_table[11].ol_amount,
                    :ord_ptr->ol_table[11].delivery_date,

                    :ord_ptr->ol_table[12].ol_i_id,
                    :ord_ptr->ol_table[12].ol_supply_w_id,
                    :ord_ptr->ol_table[12].ol_quantity,
                    :ord_ptr->ol_table[12].ol_amount,
                    :ord_ptr->ol_table[12].delivery_date,

                    :ord_ptr->ol_table[13].ol_i_id,
                    :ord_ptr->ol_table[13].ol_supply_w_id,
                    :ord_ptr->ol_table[13].ol_quantity,
                    :ord_ptr->ol_table[13].ol_amount,
                    :ord_ptr->ol_table[13].delivery_date,

                    :ord_ptr->ol_table[14].ol_i_id,
                    :ord_ptr->ol_table[14].ol_supply_w_id,
                    :ord_ptr->ol_table[14].ol_quantity,
                    :ord_ptr->ol_table[14].ol_amount,
                    :ord_ptr->ol_table[14].delivery_date

        USING

                    :ord_ptr->byname,
                    :ord_ptr->w_id,
                    :ord_ptr->d_id,
                    :ord_ptr->c_id,
                    :ord_ptr->c_last;

        ord_ptr->header.sql_code = sql_code;
        ord_ptr->header.isam_code = isam_code;

        if ( sql_code == 0 ) {
                    ord_ptr->header.returncode =
COMMIT_ORDERSTAT;
                    return 0;
        }

        if ((isam_code == DEADLOCK) || (isam_code ==
DEADLOCK_TIMEOUT )) {

                    sleep(1);
                    retry++;
                    sprintf( msg,"Order Status Transaction
Retry %d\n",retry );
                    LOGMSG( msg )
                    goto deadlock_retry;

        } else {
                    sprintf(msg, "ORDER STATUS Transaction
Fail SQL %d ISAM %d",
                                SQCODE,ISCODE);
                    LOGMSG( msg )
                    sprintf(msg, "\tw_id= %d - d_id = %d - c_id
= %d c_last = %s",
                                ord_ptr->w_id, ord_ptr->d_id,
ord_ptr->c_id, ord_ptr->c_last );
                    LOGMSG( msg )
                    sprintf(msg, "\tProcedure return SQL Code:
%d ISAM Code %d\n",
                                sql_code, isam_code);
                    LOGMSG( msg )
                    if ( SQCODE == 100 )  {
                                sprintf(msg,"ORDERSTATUS
NOTFOUND w_id=%d, d_id=%d, c_id=%d\n",
                                            ord_ptr->w_id, ord_ptr-
>d_id, ord_ptr->c_id);
                                LOGMSG( msg )
                    }
                    ord_ptr->header.returncode =
SQL_ERROR;
                    return SQL_ERROR;
        }
}

/* Stock Level Transaction */

static      int
ESQL_STKL_SVC(stkl_ptr)
$parameter register struct STKL_DATA *stkl_ptr;
{

        EXEC SQL BEGIN DECLARE SECTION;
          int sql_code,isam_code;
        EXEC SQL END DECLARE SECTION;
        int   retry;

        retry = 0;

deadlock_retry:

        EXEC SQL  EXECUTE sstocklev INTO
                    :sql_code,
                    :isam_code,
                    :stkl_ptr->stock_count
        USING
                    :stkl_ptr->w_id,
                    :stkl_ptr->stkl_d_id,
                    :stkl_ptr->threshold;

        stkl_ptr->header.sql_code = sql_code;
        stkl_ptr->header.isam_code = isam_code;

        if ( sql_code == 0 ) {
                    stkl_ptr->header.returncode =
COMMIT_STOCKLEV;
                    return 0;
        }

        if ((isam_code == DEADLOCK) || (isam_code ==
DEADLOCK_TIMEOUT )) {
                    sleep(1);
                    retry++;
                    sprintf( msg,"STOCK LEVEL Transaction
Retry %d\n",retry );
                    LOGMSG( msg )
                    goto deadlock_retry;
        } else {
                    sprintf(msg,"STOCK LEVEL Transaction
Fail SQL %d ISAM %d",SQCODE,ISCODE);
                    LOGMSG( msg )
                    sprintf(msg, "\tw_id= %d - stkl_d_id = %d -
threshold = %d",
                                stkl_ptr->w_id, stkl_ptr-
>stkl_d_id, stkl_ptr->threshold);
                    LOGMSG( msg )
                    sprintf(msg, "\tProcedure return SQL Code:
%d ISAM Code %d\n",
                                sql_code, isam_code);
                    LOGMSG( msg )
                    stkl_ptr->header.returncode =
```

```
SQL_ERROR;
                return SQL_ERROR;
        }

}

/* Delivery Transaction */
static  int
ESQL_DVRY_SVC(dvry_ptr)
$parameter register struct DVRY_DATA *dvry_ptr;
{

        EXEC SQL BEGIN DECLARE SECTION;
        short d_id;
        int   isam_code;
        int   sql_code;
        int   order_id[D_PER_W+1];
        EXEC SQL END DECLARE SECTION;

        struct  timeval end_queue, end_process;
        struct  timezone tp;
        int     i;
        int     skipped_delivery=0;
        char    dvry_log_msg[MAX_DVRY_LOG_MSG+1];
        char    *strptr;

        delivery_trans_cnt++;

        gettimeofday(&end_queue, &tp);
        dtcurrent(&infx_time);

        /*      VL      */
        dvry_ptr->header.returncode = COMMIT_DELIVERY;
        /*      VL      */

#ifdef MULTITRIP
        while ( d_id<=D_PER_W ) {

                EXEC SQL EXECUTE sxdelivery INTO
                        :order_id[d_id],
                        :sql_code,
                        :isam_code
                USING
                        :d_id,
                        :dvry_ptr->w_id,
                        :dvry_ptr->carrier_id,
                        :infx_time;

                if ( isam_code == DEADLOCK ) {
                        sprintf(msg,
                                "DELIVERY Transac-
tion Fail SQL %d ISAM %d DTC %d WID %d DID %d\n",

sql_code,isam_code,delivery_trans_cnt,dvry_ptr->w_id, d_id );
                        LOGMSG( msg )
                        sleep(delivery_trans_cnt % 3);
                        continue;
                }

                if ( sql_code == 0 ) {
                        d_id++;
                }
                else {
                        sprintf(msg,
                                "DELIVERY Transac-
tion Fail SQL %d ISAM %d DTC %d CARID %d WID %d DID
%d\n",
sql_code,isam_code, delivery_trans_cnt,
                                                dvry_ptr-
>carrier_id, dvry_ptr->w_id, d_id );
                        LOGMSG( msg )
                        d_id++;
                        /*      VL      */
                        /* Actually, we should even stop
right there */
                        dvry_ptr->header.returncode =
SQL_ERROR;
                        /*      VL      */
                }

        } /* End While */
#else
        EXEC SQL EXECUTE sdelivery INTO
                :order_id[1],
                :order_id[2],
                :order_id[3],
                :order_id[4],
                :order_id[5],
                :order_id[6],
                :order_id[7],
                :order_id[8],
                :order_id[9],
                :order_id[10],
                :sql_code,
                :isam_code
        USING
                :dvry_ptr->w_id,
                :dvry_ptr->carrier_id,
                :infx_time;

        for( d_id=1;d_id<=D_PER_W;d_id++ )
                if ( order_id[d_id] == -100 )
                        skipped_delivery++;

        if ( sql_code != 0 ) {
                sprintf(msg, "DELIVERY Transaction Fail
SQL %d ISAM %d", SQCODE, ISCODE);
                LOGMSG( msg )
                sprintf(msg, "\tDTC %d carr_id= %d w_id=
%d d_id= %d skiped= %d",
                                        delivery_trans_cnt,
dvry_ptr->carrier_id, dvry_ptr->w_id, d_id, skipped_delivery );
                LOGMSG( msg )
                sprintf(msg, "\tProcedure return SQL Code:
%d ISAM Code %d\n", sql_code, isam_code);
                LOGMSG( msg )
                /*      VL      */
                /* Actually, we should even stop right there
*/
                dvry_ptr->header.returncode =
SQL_ERROR;
                /*      VL      */
        }
#endif  /* MULTITRIP */

        gettimeofday(&end_process, &tp);


        strptr = dvry_log_msg;
        i = sprintf(strptr, "%4d %3d %d\t%d\t%d\t%4d %2d ",
                                        delivery_trans_cnt,
skipped_delivery, dvry_ptr->startq_sec,
                                        end_queue.tv_sec,
```

```
end_process.tv_sec, dvry_ptr->w_id,
                                        dvry_ptr->carrier_id);
        strptr += i;

        for (d_id=1; d_id<=D_PER_W; d_id++ )  {
                i = sprintf(strptr, "%2d %8d ", d_id,
order_id[d_id] );
                strptr += i;
        }

        *strptr++ = '\n';

        /* write all of the characters converted to the delivery
log */
        write(dvry_log, dvry_log_msg, strptr - dvry_log_msg);

        if ( dvry_ptr->header.returncode == SQL_ERROR )
                return SQL_ERROR;
        else
                return 0;

}

/
*****************************************************************************/
/
*****************************************************************************/
static void
OpenDeliveryLog()
{
        char logname[MAX_STR_LEN];
        char *envptr;

        if ( (envptr = getenv("DELIVERYLOG")) == NULL ) {
                LOGMSG( "getenv(): DELIVERYLOG not
set!\n");
                exit(99);
        }

        sprintf(logname, "%s/DVRY_LOG.%d", envptr,
getpid());
        dvry_log = creat( logname, 0666 );
        if ( dvry_log < 0 )
                LOGMSG("Failed To Open Delivery Local
Log\n")

        LOGMSG2("Opened DELIVERY log %s\n", logname)
}

static void
CloseDeliveryLog()
{
        if ( dvry_log >= 0 )
                close( dvry_log );

}

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

static void
server_loop()
{
        int             res;
        double    pdata[(MAX_DATA_LEN/sizeof(double)) +
1];
        char      status_msg[90];

        /* service people forever */
        while(1)
        {
                /* Wait for input*/
                res = read(INPUT, (void *)pdata, data_len);
                if (res == data_len)
                {
/*
                        EXEC SQL SET EXPLAIN ON;
*/
                        service_routine((void *)pdata, 0);
                }
                else if (res == 0)
                {
                        /* end of file, so quit */
                        LOGMSG( "Exiting:  Server
received EOF, quitting.\n")
                        exit(0);
                }
                else if (res == -1 )
                {
                        /* error */
                        LOGMSG2( "Exiting:  Read
Failed!  (errno = %d)\n", errno)
                        exit(3);
                }
                else
                {
                        /* error, did not read as much as
we wanted to */
                        sprintf(status_msg, "Exiting:
Read Failed (%d). Only read %d out of %d bytes\n",
                                                errno, res,
data_len);
                        LOGMSG( status_msg)
                        exit(4);
                }

                /* Write back results of retrieval */
                res = write(OUTPUT, (void *)pdata,
data_len);
                if (res == -1)
                {
                        sprintf(status_msg, "Exiting:
Write Failed (errno = %d)\n", errno);
                        LOGMSG( status_msg)
                        exit(5);
                }
        }
}
/
*****************************************************************************/
/*
                MAIN
                                        */
/
*****************************************************************************/
void
main(argc, argv)
int argc;
char **argv;
{
        char *proc_name;
        int  svr_id = -1;
```

```
        int  c, server_type;
        char    *ClientHostName = NULL;



        proc_name = argv[0];
        /*      Search for the options "-n" and "-c" from
"CLOPT"
         *      which are specified in UBBconfig
         */
        while ((c = getopt( argc, argv, "n:c:")) != EOF)
        {
                switch (c)
                {
                        case 'n':
                                svr_id = atoi(optarg);
                                break;
                        case 'c':
                                ClientHostName =
optarg;
                                break;
                }
        }

        if ( ClientHostName == NULL )
        {
                OPENLOG(999, "Error");
                LOGMSG( "Can't find the required option
'— -c' in CLOPT\n");
                exit(9);
        }

        if ( svr_id < 0 )
        {
                OPENLOG(999, ClientHostName);
                LOGMSG( "Can't find the required option
'— -n' in CLOPT\n");
                exit(9);
        }
        else
                OPENLOG(svr_id, ClientHostName);

   sleep(10);
        if (!strcmp(proc_name, "dvry_server"))
        {
                data_len        = sizeof(struct
DVRY_DATA);
                service_routine = ESQL_DVRY_SVC;
                server_type     = DELIVERY;
        }
        else if (!strcmp(proc_name, "newo_server"))
        {
                data_len        = sizeof(struct
NEWO_DATA);
                service_routine = ESQL_NEWO_SVC;
                server_type     = NEW_ORDER;
        }
        else if (!strcmp(proc_name, "pmt_server"))
        {
                data_len        = sizeof(struct
PMT_DATA);
                service_routine = ESQL_PMT_SVC;
                server_type     = PAYMENT;
        }
        else if (!strcmp(proc_name, "ords_server"))
        {
                data_len        = sizeof(struct
ORDS_DATA);
                service_routine = ESQL_ORDS_SVC;
                server_type     = ORDER_STATUS;
        }
        else if (!strcmp(proc_name, "stck_server"))
        {
                data_len        = sizeof(struct
STKL_DATA);
                service_routine = ESQL_STKL_SVC;
                server_type     = STOCK_LEVEL;
        }
        else
        {
                LOGMSG2("Error: SUT unknown server
type = %s\n", proc_name)
                exit(1);
        }

        /* attach to the database */
        if ( ESQL_init( server_type ) )
        {
                LOGMSG("TPC server FAILED to start\n")
                exit(2);
        }

        LOGMSG("TPC server has started successfully\n")

        LOGMSG("SUT Ready to receive data.\n")

        server_loop();
}
```

ex_trans.h

```
#ifndef __EX_TRANS_H__
#define __EX_TRANS_H__

#define I_NAME_LEN      24
#define I_DATA          50
#define W_NAME_LEN      10
#define ADDR_LEN        20
#define STATE_LEN       2
#define ZIP_LEN         9
#define DIST_INFO_LEN   24
#define S_DATA_LEN      50
#define D_NAME_LEN      10
#define H_DATA_LEN      24
#define CARRIER_LEN     2

#define C_LAST_LEN      17
#define C_MID_LEN       2
#define PHONE_LEN       16
#define CREDIT_LEN      2

#define C_DATA_LEN  500
#define BC_DATA_LEN  23

#define DATETIME_LEN    19

#define MAX_OL          15
```

```
#define C_LAST_LEN          17

#endif
```

neworder.sql

```
SET OPTIMIZATION HIGH;

BEGIN WORK;

DROP PROCEDURE neworder;

CREATE PROCEDURE neworder (
ol_cnt SMALLINT,
i_id1 INT, s_w_id1 SMALLINT, ol_qty1 SMALLINT,
i_id2 INT, s_w_id2 SMALLINT, ol_qty2 SMALLINT,
i_id3 INT, s_w_id3 SMALLINT, ol_qty3 SMALLINT,
i_id4 INT, s_w_id4 SMALLINT, ol_qty4 SMALLINT,
i_id5 INT, s_w_id5 SMALLINT, ol_qty5 SMALLINT,
i_id6 INT, s_w_id6 SMALLINT, ol_qty6 SMALLINT,
i_id7 INT, s_w_id7 SMALLINT, ol_qty7 SMALLINT,
i_id8 INT, s_w_id8 SMALLINT, ol_qty8 SMALLINT,
i_id9 INT, s_w_id9 SMALLINT, ol_qty9 SMALLINT,
i_id10 INT, s_w_id10 SMALLINT, ol_qty10 SMALLINT,
i_id11 INT, s_w_id11 SMALLINT, ol_qty11 SMALLINT,
i_id12 INT, s_w_id12 SMALLINT, ol_qty12 SMALLINT,
i_id13 INT, s_w_id13 SMALLINT, ol_qty13 SMALLINT,
i_id14 INT, s_w_id14 SMALLINT, ol_qty14 SMALLINT,
i_id15 INT, s_w_id15 SMALLINT, ol_qty15 SMALLINT,
warehouse_id SMALLINT,
district_id SMALLINT,
cust_id INT,
order_entry_d DATETIME YEAR TO SECOND)
RETURNING
INT,INT,INT,NUMERIC(4,4),NUMERIC(4,4),NUMERIC(4,4),CHAR(17),CHAR(3),NUMERIC(12,2),
        NUMERIC(6,2), NUMERIC(5,2), SMALLINT, CHAR(24),
CHAR(1),
        NUMERIC(6,2), NUMERIC(5,2), SMALLINT, CHAR(24),
CHAR(1),
        NUMERIC(6,2), NUMERIC(5,2), SMALLINT, CHAR(24),
CHAR(1),
        NUMERIC(6,2), NUMERIC(5,2), SMALLINT, CHAR(24),
CHAR(1),
        NUMERIC(6,2), NUMERIC(5,2), SMALLINT, CHAR(24),
CHAR(1),
        NUMERIC(6,2), NUMERIC(5,2), SMALLINT, CHAR(24),
CHAR(1),
        NUMERIC(6,2), NUMERIC(5,2), SMALLINT, CHAR(24),
CHAR(1),
        NUMERIC(6,2), NUMERIC(5,2), SMALLINT, CHAR(24),
CHAR(1),
        NUMERIC(6,2), NUMERIC(5,2), SMALLINT, CHAR(24),
CHAR(1),
        NUMERIC(6,2), NUMERIC(5,2), SMALLINT, CHAR(24),
CHAR(1),
        NUMERIC(6,2), NUMERIC(5,2), SMALLINT, CHAR(24),
CHAR(1),
        NUMERIC(6,2), NUMERIC(5,2), SMALLINT, CHAR(24),
CHAR(1),
        NUMERIC(6,2), NUMERIC(5,2), SMALLINT, CHAR(24),
CHAR(1),
        NUMERIC(6,2), NUMERIC(5,2), SMALLINT, CHAR(24),
CHAR(1),
        NUMERIC(6,2), NUMERIC(5,2), SMALLINT, CHAR(24),
CHAR(1);

DEFINE all_local, o_count, ol_num, stock_w_id, i,
stock_ord_cnt, stock_remote SMALLINT;
DEFINE ol_qty, stock_qty SMALLINT;
DEFINE item_id, stock_ytd, order_id INT;
DEFINE sql_err, isam_err INT;
DEFINE s_dist_info1, s_dist_info2, s_dist_info3, s_dist_info4,
s_dist_info5,
      s_dist_info6, s_dist_info7, s_dist_info8, s_dist_info9,
s_dist_info10,
      save_dist_info, item_name CHAR(24);
DEFINE item_data, stock_data CHAR(50);
DEFINE brand_generic CHAR(1);
DEFINE district_tax, warehouse_tax, discount NUMERIC(4,4);
DEFINE total_amt NUMERIC(12,2);
DEFINE item_price NUMERIC(5,2);
DEFINE cust_last CHAR(16);
DEFINE cust_credit CHAR(3);
DEFINE
ol_amt1,ol_amt2,ol_amt3,ol_amt4,ol_amt5,ol_amt6,ol_amt7,ol_amt8
NUMERIC(6,2);
DEFINE
ol_amt9,ol_amt10,ol_amt11,ol_amt12,ol_amt13,ol_amt14,ol_amt15,ol_amt
NUMERIC(6,2);
DEFINE  price1,price2,price3,price4,price5,price6,price7,price8
NUMERIC(5,2);
DEFINE  price9,price10,price11,price12,price13,price14,price15
NUMERIC(5,2);
DEFINE
s_qty1,s_qty2,s_qty3,s_qty4,s_qty5,s_qty6,s_qty7,s_qty8
SMALLINT;
DEFINE
s_qty9,s_qty10,s_qty11,s_qty12,s_qty13,s_qty14,s_qty15
SMALLINT;
DEFINE
iname1,iname2,iname3,iname4,iname5,iname6,iname7,iname8
CHAR(24);
DEFINE
iname9,iname10,iname11,iname12,iname13,iname14,iname15
CHAR(24);
DEFINE
brand1,brand2,brand3,brand4,brand5,brand6,brand7,brand8
CHAR(1);
DEFINE
brand9,brand10,brand11,brand12,brand13,brand14,brand15
CHAR(1);
DEFINE d_info1, d_info2, d_info3, d_info4, d_info5, d_info6,
d_info7, d_info8,
      d_info9, d_info10, d_info11, d_info12, d_info13, d_info14,
d_info15
      CHAR(24);

ON EXCEPTION SET sql_err, isam_err
            ROLLBACK WORK;
            RETURN sql_err, isam_err, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;
END EXCEPTION;
```

```
— SET DEBUG FILE TO '/tmp/neworder.trc';
— TRACE ON;
— SET EXPLAIN ON;

LET o_count = 0;

LET all_local, ol_num, i = 1, 1, 1;
LET
ol_amt1,ol_amt2,ol_amt3,ol_amt4,ol_amt5,ol_amt6,ol_amt7,ol_amt8=0,0,0,0,0,0,0,0;
LET
ol_amt9,ol_amt10,ol_amt11,ol_amt12,ol_amt13,ol_amt14,ol_amt15
= 0,0,0,0,0,0,0;
LET price1,price2,price3,price4,price5,price6,price7 ,price8 =
0,0,0,0,0,0,0,0;
LET price9,price10,price11,price12,price13,price14,price15 =
0,0,0,0,0,0,0;
LET  s_qty1,s_qty2,s_qty3,s_qty4,s_qty5,s_qty6,s_qty7,s_qty8
= 0,0,0,0,0,0,0,0;
LET  s_qty9,s_qty10,s_qty11,s_qty12,s_qty13,s_qty14,s_qty15
= 0,0,0,0,0,0,0;
LET
iname1,iname2,iname3,iname4,iname5,iname6,iname7,iname8
= 0,0,0,0,0,0,0,0;
LET
iname9,iname10,iname11,iname12,iname13,iname14,iname15
= 0,0,0,0,0,0,0;
LET
brand1,brand2,brand3,brand4,brand5,brand6,brand7,brand8 =
0,0,0,0,0,0,0,0;
LET
brand9,brand10,brand11,brand12,brand13,brand14,brand15 =
0,0,0,0,0,0,0;
LET total_amt = 0;

— TRACE warehouse_id;
— TRACE district_id;

FOREACH cur1 FOR SELECT d_next_o_id, district.d_tax
          INTO order_id, district_tax
          FROM district
          WHERE district.d_id = district_id AND d_w_id =
warehouse_id

          UPDATE district SET d_next_o_id = order_id + 1
WHERE CURRENT OF cur1;
          — TRACE order_id;
          — TRACE district_tax;
END FOREACH;

WHILE ol_num <= ol_cnt

IF i < 9 THEN                  — 1 2 3 4 5 6 7 8
   IF i < 5 THEN               — 1 2 3 4
      IF i < 3 THEN            — 1 2
         IF i < 2 THEN         — 1
            LET item_id, stock_w_id, ol_qty = i_id1, s_w_id1,
ol_qty1;
         ELSE                  — 2
            LET item_id, stock_w_id, ol_qty = i_id2, s_w_id2,
ol_qty2;
         END IF;
      ELSE                     — 3 4
         IF i < 4 THEN         — 3
            LET item_id, stock_w_id, ol_qty = i_id3, s_w_id3,
ol_qty3;
         ELSE                  — 4
            LET item_id, stock_w_id, ol_qty = i_id4, s_w_id4,
ol_qty4;
         END IF;
      END IF;
   ELSE                        — 5 6 7 8
      IF i < 7 THEN            — 5 6
         IF i < 6 THEN         — 5
            LET item_id, stock_w_id, ol_qty = i_id5, s_w_id5,
ol_qty5;
         ELSE                  — 6
            LET item_id, stock_w_id, ol_qty = i_id6, s_w_id6,
ol_qty6;
         END IF;
      ELSE                     — 7 8
         IF I < 8 THEN         — 7
            LET item_id, stock_w_id, ol_qty = i_id7, s_w_id7,
ol_qty7;
         ELSE                  — 8
            LET item_id, stock_w_id, ol_qty = i_id8, s_w_id8,
ol_qty8;
         END IF;
      END IF;
   END IF;
ELSE                           — 9 10 11 12 13 14 15
   IF i < 13 THEN              — 9 10 11 12
      IF i < 11 THEN           — 9 10
         IF i < 10 THEN        — 9
            LET item_id, stock_w_id, ol_qty = i_id9, s_w_id9,
ol_qty9;
         ELSE                  — 10
            LET item_id, stock_w_id, ol_qty = i_id10, s_w_id10,
ol_qty10;
         END IF;
      ELSE                     — 11 12
         IF i < 12 THEN        — 11
            LET item_id, stock_w_id, ol_qty = i_id11, s_w_id11,
ol_qty11;
         ELSE                  — 12
            LET item_id, stock_w_id, ol_qty = i_id12, s_w_id12,
ol_qty12;
         END IF;
      END IF;
   ELSE                        — 13 14 15
      IF i < 15 THEN           — 13 14
         IF i < 14 THEN        — 13
            LET item_id, stock_w_id, ol_qty = i_id13, s_w_id13,
ol_qty13;
         ELSE                  — 14
            LET item_id, stock_w_id, ol_qty = i_id14, s_w_id14,
ol_qty14;
         END IF;
      ELSE                     — 15
         LET item_id, stock_w_id, ol_qty = i_id15, s_w_id15,
ol_qty15;
      END IF;
   END IF;
END IF;

SELECT i_price, i_name, i_data
          INTO item_price, item_name, item_data
          FROM item
          WHERE i_id = item_id;

— TRACE item_price;
— TRACE item_name;
— TRACE item_data;
IF item_name IS NULL AND item_price IS NULL AND
```

```
item_data IS NULL THEN
        — RAISE EXCEPTION 100;
        EXIT WHILE;
END IF;

FOREACH cur2 FOR SELECT s_quantity, s_data, s_ytd,
s_order_cnt, s_remote_cnt,
                s_dist_01, s_dist_02, s_dist_03, s_dist_04,
s_dist_05, s_dist_06,
                s_dist_07, s_dist_08, s_dist_09, s_dist_10
        INTO stock_qty, stock_data, stock_ytd,
stock_ord_cnt, stock_remote,
                s_dist_info1, s_dist_info2, s_dist_info3,
s_dist_info4,
                s_dist_info5, s_dist_info6, s_dist_info7,
s_dist_info8,
                s_dist_info9, s_dist_info10
        FROM stock
        WHERE s_i_id = item_id AND s_w_id = stock_w_id

        LET stock_qty = stock_qty - ol_qty;

        IF stock_qty < 10 THEN
                LET stock_qty = stock_qty + 91;
        END IF;

        LET stock_ytd = stock_ytd + ol_qty;
        LET stock_ord_cnt = stock_ord_cnt + 1;

        IF stock_w_id <> warehouse_id THEN
                LET stock_remote = stock_remote + 1;
                LET all_local = 0;
        END IF;

        UPDATE stock
        SET s_quantity = stock_qty, s_ytd =
stock_ytd,s_order_cnt = stock_ord_cnt,
                s_remote_cnt = stock_remote
        WHERE CURRENT OF cur2;
END FOREACH;

IF district_id < 5 THEN                         — 1 2 3 4
   IF district_id < 3 THEN                      — 1 2
      IF district_id < 2 THEN                   — 1
         LET save_dist_info = s_dist_info1;
      ELSE                                      — 2
         LET save_dist_info = s_dist_info2;
      END IF;
   ELSE                                         — 3 4
      IF district_id < 4 THEN                   — 3
         LET save_dist_info = s_dist_info3;
      ELSE                                      — 4
         LET save_dist_info = s_dist_info4;
      END IF;
   END IF;
ELSE                                            — 5 6 7 8 9 10
   IF district_id < 7 THEN                      — 5 6
      IF district_id < 6 THEN                   — 5
         LET save_dist_info = s_dist_info5;
      ELSE                                      — 6
         LET save_dist_info = s_dist_info6;
      END IF;
   ELSE                                         — 7 8 9 10
      IF district_id < 9 THEN                   — 7 8
         IF district_id < 8 THEN                — 7
            LET save_dist_info = s_dist_info7;
         ELSE                                   — 8
            LET save_dist_info = s_dist_info8;
         END IF;
      ELSE                                      — 9 10
         IF district_id < 10 THEN               — 9
            LET save_dist_info = s_dist_info9;
         ELSE                                   — 10
            LET save_dist_info = s_dist_info10;
         END IF;
      END IF;
   END IF;
END IF;

LET ol_amt = ol_qty * item_price;

IF item_data MATCHES '*ORIGINAL*' AND stock_data
MATCHES '*ORIGINAL*' THEN
        LET brand_generic = 'B';
ELSE
        LET brand_generic = 'G';
END IF;

IF i < 9 THEN                                   — 1 2 3 4 5 6 7 8
   IF i < 5 THEN                                — 1 2 3 4
      IF i < 3 THEN                             — 1 2
         IF i < 2 THEN                          — 1
            LET ol_amt1, price1, s_qty1 = ol_amt, item_price,
stock_qty;
            LET iname1, brand1 = item_name, brand_generic;
                LET d_info1 = save_dist_info;
         ELSE                                   — 2
            LET ol_amt2, price2, s_qty2 = ol_amt, item_price,
stock_qty;
            LET iname2, brand2 = item_name, brand_generic;
                LET d_info2 = save_dist_info;
         END IF;
      ELSE                                      — 3 4
         IF i < 4 THEN                          — 3
            LET ol_amt3, price3, s_qty3 = ol_amt, item_price,
stock_qty;
            LET iname3, brand3 = item_name, brand_generic;
                LET d_info3 = save_dist_info;
         ELSE                                   — 4
            LET ol_amt4, price4, s_qty4 = ol_amt, item_price,
stock_qty;
            LET iname4, brand4 = item_name, brand_generic;
                LET d_info4 = save_dist_info;
         END IF;
      END IF;
   ELSE                                         — 5 6 7 8
      IF i < 7 THEN                             — 5 6
         IF i < 6 THEN                          — 5
            LET ol_amt5, price5, s_qty5 = ol_amt, item_price,
stock_qty;
            LET iname5, brand5 = item_name, brand_generic;
                LET d_info5 = save_dist_info;
         ELSE                                   — 6
            LET ol_amt6, price6, s_qty6 = ol_amt, item_price,
stock_qty;
            LET iname6, brand6 = item_name, brand_generic;
                LET d_info6 = save_dist_info;
         END IF;
      ELSE                                      — 7 8
         IF I < 8 THEN                          — 7
            LET ol_amt7, price7, s_qty7 = ol_amt, item_price,
stock_qty;
            LET iname7, brand7 = item_name, brand_generic;
                LET d_info7 = save_dist_info;
```

```
        ELSE                    — 8
            LET ol_amt8, price8, s_qty8 = ol_amt, item_price,
stock_qty;
            LET iname8, brand8 = item_name, brand_generic;
                LET d_info8 = save_dist_info;
        END IF;
      END IF;
   END IF;
ELSE                            — 9 10 11 12 13 14 15
   IF i < 13 THEN               — 9 10 11 12
      IF i < 11 THEN            — 9 10
         IF i < 10 THEN         — 9
            LET ol_amt9, price9, s_qty9 = ol_amt, item_price,
stock_qty;
            LET iname9, brand9 = item_name, brand_generic;
                LET d_info9 = save_dist_info;
         ELSE                   — 10
            LET ol_amt10, price10, s_qty10 =
ol_amt,item_price,stock_qty;
            LET iname10, brand10 = item_name, brand_generic;
                LET d_info10 = save_dist_info;
        END IF;
      ELSE                      — 11 12
         IF i < 12 THEN         — 11
            LET ol_amt11, price11, s_qty11 =
ol_amt,item_price,stock_qty;
            LET iname11, brand11 = item_name, brand_generic;
                LET d_info11 = save_dist_info;
         ELSE                   — 12
            LET ol_amt12, price12, s_qty12 =
ol_amt,item_price,stock_qty;
            LET iname12, brand12 = item_name, brand_generic;
                LET d_info12 = save_dist_info;
        END IF;
      END IF;
   ELSE                         — 13 14 15
      IF i < 15 THEN            — 13 14
         IF i < 14 THEN         — 13
            LET ol_amt13, price13, s_qty13 =
ol_amt,item_price,stock_qty;
            LET iname13, brand13 = item_name, brand_generic;
                LET d_info13 = save_dist_info;
         ELSE                   — 14
            LET ol_amt14, price14, s_qty14 =
ol_amt,item_price,stock_qty;
            LET iname14, brand14 = item_name, brand_generic;
                LET d_info14 = save_dist_info;
        END IF;
      ELSE                      — 15
            LET ol_amt15, price15, s_qty15 =
ol_amt,item_price,stock_qty;
            LET iname15, brand15 = item_name, brand_generic;
                LET d_info15 = save_dist_info;
      END IF;
   END IF;
END IF;

LET total_amt = total_amt + ol_amt;

LET o_count = o_count + 1;
LET ol_num = ol_num + 1;
LET i = i + 1;
END WHILE;

LET ol_num = 1;
LET i = 1;
WHILE ol_num <= o_count

IF i < 9 THEN                   — 1 2 3 4 5 6 7 8
   IF i < 5 THEN                — 1 2 3 4
      IF i < 3 THEN             — 1 2
         IF i < 2 THEN          — 1
            LET item_id, stock_w_id, ol_qty, ol_amt =
                                i_id1, s_w_id1, ol_qty1, ol_amt1;
                LET save_dist_info = d_info1;
         ELSE                   — 2
            LET item_id, stock_w_id, ol_qty, ol_amt =
                                i_id2, s_w_id2, ol_qty2, ol_amt2;
                LET save_dist_info = d_info2;
         END IF;
      ELSE                      — 3 4
         IF i < 4 THEN          — 3
            LET item_id, stock_w_id, ol_qty, ol_amt =
                                i_id3, s_w_id3, ol_qty3, ol_amt3;
                LET save_dist_info = d_info3;
         ELSE                   — 4
            LET item_id, stock_w_id, ol_qty, ol_amt =
                                i_id4, s_w_id4, ol_qty4, ol_amt4;
                LET save_dist_info = d_info4;
         END IF;
      END IF;
   ELSE                         — 5 6 7 8
      IF i < 7 THEN             — 5 6
         IF i < 6 THEN          — 5
            LET item_id, stock_w_id, ol_qty, ol_amt =
                                i_id5, s_w_id5, ol_qty5, ol_amt5;
                LET save_dist_info = d_info5;
         ELSE                   — 6
            LET item_id, stock_w_id, ol_qty, ol_amt =
                                i_id6, s_w_id6, ol_qty6, ol_amt6;
                LET save_dist_info = d_info6;
         END IF;
      ELSE                      — 7 8
         IF I < 8 THEN          — 7
            LET item_id, stock_w_id, ol_qty, ol_amt =
                                i_id7, s_w_id7, ol_qty7, ol_amt7;
                LET save_dist_info = d_info7;
         ELSE                   — 8
            LET item_id, stock_w_id, ol_qty, ol_amt =
                                i_id8, s_w_id8, ol_qty8, ol_amt8;
                LET save_dist_info = d_info8;
         END IF;
      END IF;
   END IF;
ELSE                            — 9 10 11 12 13 14 15
   IF i < 13 THEN               — 9 10 11 12
      IF i < 11 THEN            — 9 10
         IF i < 10 THEN         — 9
            LET item_id, stock_w_id, ol_qty, ol_amt =
                                i_id9, s_w_id9, ol_qty9, ol_amt9;
                LET save_dist_info = d_info9;
         ELSE                   — 10
            LET item_id, stock_w_id, ol_qty, ol_amt =
                                i_id10, s_w_id10, ol_qty10,
ol_amt10;
                LET save_dist_info = d_info10;
         END IF;
      ELSE                      — 11 12
         IF i < 12 THEN         — 11
            LET item_id, stock_w_id, ol_qty, ol_amt =
                                i_id11, s_w_id11, ol_qty11,
ol_amt11;
                LET save_dist_info = d_info11;
```

```
        ELSE                    — 12
            LET item_id, stock_w_id, ol_qty, ol_amt =
                                i_id12, s_w_id12, ol_qty12,
ol_amt12;
                    LET save_dist_info = d_info12;
        END IF;
      END IF;
    ELSE                        — 13 14 15
      IF i < 15 THEN            — 13 14
        IF i < 14 THEN          — 13
            LET item_id, stock_w_id, ol_qty, ol_amt =
                                i_id13, s_w_id13, ol_qty13,
ol_amt13;
                    LET save_dist_info = d_info13;
        ELSE                    — 14
            LET item_id, stock_w_id, ol_qty, ol_amt =
                                i_id14, s_w_id14, ol_qty14,
ol_amt14;
                    LET save_dist_info = d_info14;
        END IF;
      ELSE                      — 15
            LET item_id, stock_w_id, ol_qty, ol_amt =
                                i_id15, s_w_id15, ol_qty15,
ol_amt15;
                    LET save_dist_info = d_info15;
      END IF;
    END IF;
END IF;

— TRACE order_id;
— TRACE district_id;
— TRACE warehouse_id;
— TRACE ol_num;
— TRACE item_id;
— TRACE stock_w_id;

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 (order_id, district_id, warehouse_id,
ol_num, item_id,
                    stock_w_id, NULL, ol_qty, ol_amt,
save_dist_info);

LET ol_num = ol_num + 1;
LET i = i + 1;
END WHILE;


SELECT customer.c_last, customer.c_credit,
customer.c_discount, warehouse.w_tax
            INTO cust_last, cust_credit, discount, warehouse_tax
            FROM customer, warehouse
            WHERE c_w_id = warehouse_id AND
            c_d_id = district_id AND
            customer.c_id = cust_id AND
            warehouse.w_id = warehouse_id;

IF cust_last IS NULL THEN
            RAISE EXCEPTION 200;
END IF;

INSERT INTO order (o_id, o_d_id, o_w_id, o_c_id, o_entry_d,
o_carrier_id,
                    o_ol_cnt, o_all_local)
            VALUES (order_id, district_id, warehouse_id,
cust_id, order_entry_d,
                    0, ol_cnt, all_local);          — 0 for
NULL, cf load_tpcc.ec

INSERT INTO new_order (no_o_id, no_d_id, no_w_id)
            VALUES (order_id, district_id, warehouse_id);

LET total_amt =
            total_amt * (1.0 + warehouse_tax + district_tax) * (1.0
- discount);

IF o_count = ol_cnt THEN
            COMMIT WORK;
ELSE
            ROLLBACK WORK;
            RETURN 100, 0, order_id, 0, 0, 0, cust_last,
cust_credit, 0,
                    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;
END IF;

RETURN 0, 0, order_id, warehouse_tax, district_tax, discount,
      cust_last, cust_credit, total_amt,
      ol_amt1, price1, s_qty1, iname1, brand1,
      ol_amt2, price2, s_qty2, iname2, brand2,
      ol_amt3, price3, s_qty3, iname3, brand3,
      ol_amt4, price4, s_qty4, iname4, brand4,
      ol_amt5, price5, s_qty5, iname5, brand5,
      ol_amt6, price6, s_qty6, iname6, brand6,
      ol_amt7, price7, s_qty7, iname7, brand7,
      ol_amt8, price8, s_qty8, iname8, brand8,
      ol_amt9, price9, s_qty9, iname9, brand9,
      ol_amt10, price10, s_qty10, iname10, brand10,
      ol_amt11, price11, s_qty11, iname11, brand11,
      ol_amt12, price12, s_qty12, iname12, brand12,
      ol_amt13, price13, s_qty13, iname13, brand13,
      ol_amt14, price14, s_qty14, iname14, brand14,
      ol_amt15, price15, s_qty15, iname15, brand15;

END PROCEDURE;

COMMIT WORK;
```

payment.sql

```
SET OPTIMIZATION HIGH;

BEGIN WORK;

DROP PROCEDURE payment;

CREATE PROCEDURE payment (
did SMALLINT,                       — pmt->d_id
cid INT,                    — pmt->c_id
clast CHAR(16),                     — pmt->c_last
c_did SMALLINT,                     — pmt->c_d_id
c_wid SMALLINT,                     — pmt->c_w_id
hamount NUMERIC(12,2),                      — pmt-
>h_amount / 100
```

```
wid SMALLINT,                          — pmt->w_id
byname INT,                            — pmt->byname
hdate DATETIME YEAR TO SECOND  — pmt->pay_date
)
RETURNING
INT,       — sql_code
INT,       — isam_err
INT,       — c_id       Either c_id or c_last will be 0 on input,
CHAR(16),          — c_last[17]   and the procedure will need
to return its value
CHAR(20),          — w_street_1[21]
CHAR(20),          — w_street_2[21]
CHAR(20),          — w_city[21]
CHAR(2),— w_state[3]
CHAR(9),— w_zip[10]
CHAR(20),          — d_street_1[21]
CHAR(20),          — d_street_2[21]
CHAR(20),          — d_city[21]
CHAR(2),— d_state[3]
CHAR(9),— d_zip[10]
CHAR(16),          — c_first[17]
CHAR(2),— c_middle[3]
CHAR(20),          — c_street_1[21]
CHAR(20),          — c_street_2[21]
CHAR(20),          — c_city[21]
CHAR(2),— c_state[3]
CHAR(9),— c_zip[10]
CHAR(16),          — c_phone[17]
DATETIME YEAR TO SECOND, — c_since[20]
CHAR(2),— c_credit[3]
NUMERIC(12,2),     — c_credit_lim
NUMERIC(4,4),      — c_discount
NUMERIC(12,2),     — c_balance
CHAR(200);         — c_data[501]


DEFINE wstreet1, wstreet2, wcity, dstreet1, dstreet2, dcity,
      cstreet1, cstreet2, ccity CHAR(20);
DEFINE wstate, dstate, cmiddle, cstate, ccredit CHAR(2);
DEFINE wzip, dzip, czip CHAR(9);
DEFINE cfirst, cphone CHAR(16);
DEFINE csince CHAR(25);
DEFINE cdiscount NUMERIC(4,4);
DEFINE cbalance, ccreditlim, amt NUMERIC(12,2);
DEFINE cdata CHAR(500);
DEFINE namecount,i INT;
DEFINE wname,dname CHAR(10);
DEFINE hdata CHAR(24);
DEFINE sql_code,isam_err INT;


ON EXCEPTION SET sql_code, isam_err
   ROLLBACK WORK;
   RETURN sql_code, isam_err, cid, clast, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, NULL, 0, 0, 0, 0, 0;
END EXCEPTION;

— SET DEBUG FILE TO '/tmp/payment.trc';
— TRACE ON;
— SET EXPLAIN ON;

LET cdata=" ";

IF byname <> 0 THEN    — do payment by last name
           SELECT COUNT(*) INTO namecount FROM
customer
           WHERE c_w_id = c_wid AND c_d_id = c_did AND
c_last = clast;
           IF namecount <> 0 THEN
              LET i = 1;
              LET namecount = (namecount + 1)/ 2;
              FOREACH cur3 FOR SELECT c_id, c_first
                 INTO cid, cfirst
                 FROM customer
                 WHERE c_w_id = c_wid AND c_d_id = c_did
AND c_last = clast
                 ORDER BY c_first
                 IF i = namecount THEN
                        SELECT c_middle, c_last,
                                                  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
                           INTO   cmiddle, clast, cstreet1, cstreet2,
ccity,
                                  cstate, czip, cphone, csince,
ccredit,
                                  ccreditlim, cdiscount, cbalance
                           FROM customer
                           WHERE c_w_id = c_wid AND c_d_id =
c_did AND c_id = cid;
                           EXIT FOREACH;
                       END IF;
                       LET i = i + 1;
              END FOREACH;
           ELSE
              RAISE EXCEPTION 200; — handle error
           END IF;
ELSE                          — do payment by c_id
           SELECT c_first, c_middle, c_last, 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
           INTO cfirst, cmiddle, clast, cstreet1, cstreet2, ccity,
cstate,
                czip, cphone, csince, ccredit, ccreditlim,
cdiscount, cbalance
           FROM customer
      WHERE c_w_id = c_wid AND c_d_id = c_did AND c_id =
cid;

           IF clast IS NULL THEN
                    RAISE EXCEPTION 200;
           END IF;
END IF;

LET cbalance = cbalance - hamount;

IF ccredit = 'BC' THEN
   SELECT c_data[1,471] INTO cdata[30,500]
   FROM customer
   WHERE c_w_id = c_wid AND c_d_id = c_did AND c_id =
cid;

   LET cdata[1,5] = cid;
   LET cdata[6,8] = c_did;
   LET cdata[9,13] = c_wid;
   LET cdata[14,16] = did;
   LET cdata[17,21] = wid;
   LET cdata[22] = '$';
   LET cdata[23,29] = hamount;
```

```
   UPDATE customer SET c_data        = cdata,
                       c_balance     = cbalance,
                       c_ytd_payment = c_ytd_payment + hamount,
                       c_payment_cnt = c_payment_cnt + 1
   WHERE c_w_id = c_wid AND c_d_id = c_did AND c_id =
cid;
ELSE
   UPDATE customer
      SET c_balance     = cbalance,
          c_ytd_payment = c_ytd_payment + hamount,
          c_payment_cnt = c_payment_cnt + 1
      WHERE c_w_id = c_wid AND c_d_id = c_did AND c_id =
cid;
END IF;

SELECT d_name, d_street_1, d_street_2, d_city, d_state,
d_zip
            INTO dname, dstreet1, dstreet2, dcity, dstate, dzip
            FROM district
            WHERE d_w_id = wid AND d_id = did;

IF dname IS NULL THEN
            RAISE EXCEPTION 200;
END IF;

UPDATE district SET d_ytd = d_ytd + hamount
            WHERE d_w_id = wid AND d_id = did;

SELECT w_name, w_street_1, w_street_2, w_city, w_state,
w_zip
       INTO wname, wstreet1, wstreet2, wcity, wstate, wzip
       FROM warehouse
       WHERE w_id = wid;

IF wname IS NULL THEN
            RAISE EXCEPTION 200;
END IF;

UPDATE warehouse SET w_ytd = w_ytd + hamount
       WHERE w_id = wid;

LET hdata = wname ||'    '|| dname;
LET amt = hamount ;
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 (cid, c_did, c_wid, did, wid, hdate, amt,
hdata);

COMMIT WORK;

RETURN 0, 0, cid, clast, wstreet1, wstreet2, wcity, wstate,
      wzip, dstreet1, dstreet2, dcity, dstate, dzip, cfirst, cmiddle,
      cstreet1, cstreet2, ccity, cstate, czip, cphone, csince,
ccredit,
      ccreditlim, cdiscount, cbalance, cdata[1,200];

END PROCEDURE;

COMMIT WORK;
```

stocklev.sql

```
SET OPTIMIZATION HIGH;

BEGIN WORK;

DROP PROCEDURE stock_level;

CREATE PROCEDURE stock_level
(warehouse_id SMALLINT,
 district_id SMALLINT,
 threshold INT)
RETURNING
INT,    — sql_code
INT,    — isam_code
INT     — low_stock
;

DEFINE sql_code, isam_code INT;
DEFINE next_o_id, I, low_stock   INT;

ON EXCEPTION SET sql_code, isam_code
            ROLLBACK WORK;
            RETURN sql_code, isam_code, -1;
END EXCEPTION;

— SET DEBUG FILE TO '/tmp/stocklev.trc';
— TRACE ON;
— SET EXPLAIN ON;

SELECT d_next_o_id INTO next_o_id FROM district
            WHERE d_w_id = warehouse_id AND d_id =
district_id;

FOR I in ((next_o_id-1) to (next_o_id-20))
            INSERT INTO stk_lvl_tmp
                        SELECT s_i_id FROM order_line, stock
                        WHERE ol_w_id = warehouse_id AND
                                    ol_d_id = district_id AND
                                    ol_o_id = I AND
                                    s_i_id = ol_i_id AND s_w_id =
ol_w_id AND
                                    s_quantity < threshold;
END FOR


SELECT COUNT ( DISTINCT t_i_id) INTO low_stock FROM
stk_lvl_tmp;

DELETE FROM stk_lvl_tmp;

COMMIT WORK;

RETURN 0, 0, low_stock;

END PROCEDURE;

COMMIT WORK;
```

ordstat.sql

```
SET OPTIMIZATION HIGH;

BEGIN WORK;

DROP PROCEDURE order_status;

CREATE PROCEDURE order_status
(byname SMALLINT,          — ord_ptr->byname
 warehouse_id SMALLINT, — ord_ptr->w_id
 district_id SMALLINT,  — ord_ptr->d_id
 customer_id INT,       — ord_ptr->c_id
 customer_last CHAR(16) — ord_ptr->c_last
)
RETURNING
INT,                              — sql_code
INT,                              — isam_code
INT,                              — c_id
CHAR(17),                         — c_first
CHAR(3),                   — c_middle
CHAR(17),                         — c_last
NUMERIC(12,2),                    — c_balance
INT,                              — o_id
DATETIME YEAR TO SECOND,          — entry_d
SMALLINT,                         — carrier_id
SMALLINT,                         — ol_cnt
INT,SMALLINT,SMALLINT,NUMERIC(6,2),DATETIME YEAR
TO SECOND,
INT,SMALLINT,SMALLINT,NUMERIC(6,2),DATETIME YEAR
TO SECOND,
INT,SMALLINT,SMALLINT,NUMERIC(6,2),DATETIME YEAR
TO SECOND,
INT,SMALLINT,SMALLINT,NUMERIC(6,2),DATETIME YEAR
TO SECOND,
INT,SMALLINT,SMALLINT,NUMERIC(6,2),DATETIME YEAR
TO SECOND,
INT,SMALLINT,SMALLINT,NUMERIC(6,2),DATETIME YEAR
TO SECOND,
INT,SMALLINT,SMALLINT,NUMERIC(6,2),DATETIME YEAR
TO SECOND,
INT,SMALLINT,SMALLINT,NUMERIC(6,2),DATETIME YEAR
TO SECOND,
INT,SMALLINT,SMALLINT,NUMERIC(6,2),DATETIME YEAR
TO SECOND,
INT,SMALLINT,SMALLINT,NUMERIC(6,2),DATETIME YEAR
TO SECOND,
INT,SMALLINT,SMALLINT,NUMERIC(6,2),DATETIME YEAR
TO SECOND,
INT,SMALLINT,SMALLINT,NUMERIC(6,2),DATETIME YEAR
TO SECOND,
INT,SMALLINT,SMALLINT,NUMERIC(6,2),DATETIME YEAR
TO SECOND,
INT,SMALLINT,SMALLINT,NUMERIC(6,2),DATETIME YEAR
TO SECOND,
INT,SMALLINT,SMALLINT,NUMERIC(6,2),DATETIME YEAR
TO SECOND;
                                  —
i_id,supplay_w_id,quantity,amount,delivery_d

DEFINE sql_code, isam_code, i INT;
DEFINE namecnt, ol_cnt , carrier_id SMALLINT;
DEFINE cust_first, cust_last CHAR(17);
DEFINE cust_middle CHAR(3);
DEFINE cust_balance NUMERIC(12,2);
DEFINE order_id INT;
DEFINE entry_d DATETIME YEAR TO SECOND;

DEFINE i_id1,i_id2,i_id3,i_id4,i_id5,i_id6,i_id7,i_id8 INT;
DEFINE i_id9,i_id10,i_id11,i_id12,i_id13,i_id14,i_id15, i_id INT;
DEFINE
supply_w_id1,supply_w_id2,supply_w_id3,supply_w_id4,supply_w_id5
SMALLINT;
DEFINE
supply_w_id6,supply_w_id7,supply_w_id8,supply_w_id9,supply_w_id10
SMALLINT;
DEFINE
supply_w_id11,supply_w_id12,supply_w_id13,supply_w_id14
SMALLINT;
DEFINE supply_w_id15,supply_w_id SMALLINT;
DEFINE
quantity1,quantity2,quantity3,quantity4,quantity5,quantity6
SMALLINT;
DEFINE
quantity7,quantity8,quantity9,quantity10,quantity11,quantity12
SMALLINT;
DEFINE quantity13,quantity14,quantity15, quantity SMALLINT;
DEFINE
amount1,amount2,amount3,amount4,amount5,amount6,amount7,amount8
NUMERIC(6,2);
DEFINE
amount9,amount10,amount11,amount12,amount13,amount14,amount15,amount
NUMERIC(6,2);
DEFINE delivery_d1,delivery_d2,delivery_d3,delivery_d4
DATETIME YEAR TO SECOND;
DEFINE delivery_d5,delivery_d6,delivery_d7,delivery_d8
DATETIME YEAR TO SECOND;
DEFINE delivery_d9,delivery_d10,delivery_d11 DATETIME
YEAR TO SECOND;
DEFINE delivery_d12,delivery_d13,delivery_d14 DATETIME
YEAR TO SECOND;
DEFINE delivery_d15,delivery_d DATETIME YEAR TO
SECOND;

ON EXCEPTION SET sql_code, isam_code
        ROLLBACK WORK;
        RETURN sql_code, isam_code,
0,0,0,0,0,0,NULL,0,0,

0,0,0,0,NULL,0,0,0,0,NULL,0,0,0,0,NULL,0,0,0,0,NULL,

0,0,0,0,NULL,0,0,0,0,NULL,0,0,0,0,NULL,0,0,0,0,NULL,

0,0,0,0,NULL,0,0,0,0,NULL,0,0,0,0,NULL,0,0,0,0,NULL,
                0,0,0,0,NULL,0,0,0,0,NULL,0,0,0,0,NULL;
END EXCEPTION;

— SET DEBUG FILE TO ‘/tmp/ordstat.trc’;
— TRACE ON;
— SET EXPLAIN ON;

LET i_id1,i_id2,i_id3,i_id4,i_id5,i_id6,i_id7,i_id8
=0,0,0,0,0,0,0,0;
LET i_id9,i_id10,i_id11,i_id12,i_id13,i_id14,i_id15
=0,0,0,0,0,0,0;
LET
supply_w_id1,supply_w_id2,supply_w_id3,supply_w_id4,supply_w_id5
        =0,0,0,0,0;
LET
supply_w_id6,supply_w_id7,supply_w_id8,supply_w_id9,supply_w_id10
        = 0,0,0,0,0;
```

```
LET
supply_w_id11,supply_w_id12,supply_w_id13,supply_w_id14
=0,0,0,0;
LET supply_w_id15 =0;
LET  quantity1,quantity2,quantity3,quantity4,quantity5,quantity6
          = 0,0,0,0,0,0;
LET
quantity7,quantity8,quantity9,quantity10,quantity11,quantity12
          = 0,0,0,0,0,0;
LET quantity13,quantity14,quantity15 = 0,0,0;
LET
amount1,amount2,amount3,amount4,amount5,amount6,amount7,amount8
          =  0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0;
LET
amount9,amount10,amount11,amount12,amount13,amount14,amount15
          =  0.0,0.0,0.0,0.0,0.0,0.0,0.0;
LET delivery_d1,delivery_d2,delivery_d3,delivery_d4
          = NULL, NULL, NULL, NULL;
LET delivery_d5,delivery_d6,delivery_d7,delivery_d8
          = NULL, NULL, NULL, NULL;
LET delivery_d9,delivery_d10, delivery_d11,delivery_d12
          = NULL, NULL, NULL, NULL;
LET delivery_d13,delivery_d14,delivery_d15
          = NULL, NULL, NULL;

IF byname <> 0 THEN   — by last name
   SELECT COUNT(*) INTO namecnt FROM customer
          WHERE  c_w_id = warehouse_id AND
               c_d_id = district_id AND
               c_last = customer_last;

   IF namecnt = 0 THEN
                    RAISE EXCEPTION 200;
   ELSE
                    LET namecnt = (namecnt + 1) / 2 ;
                    LET i = 1;

                    FOREACH cur3 FOR SELECT c_id, c_first
                    INTO customer_id, cust_first
                    FROM customer
                    WHERE  c_w_id = warehouse_id AND
                    c_d_id = district_id AND
                    c_last = customer_last
                    ORDER BY c_first

                    IF i = namecnt THEN
                              SELECT c_balance, c_last,
c_middle
                              INTO cust_balance, cust_last,
cust_middle
                              FROM customer
                              WHERE c_w_id = warehouse_id
AND
                                                  c_d_id =
district_id AND
                                                  c_id =
customer_id;
                    EXIT FOREACH;
          END IF;

          LET i = i + 1;

          END FOREACH;
   END IF;
ELSE  — by c_id
   SELECT c_balance, c_first, c_middle, c_last
          INTO cust_balance, cust_first, cust_middle, cust_last
          FROM customer
          WHERE c_id = customer_id AND
               c_w_id = warehouse_id AND
               c_d_id = district_id;
          IF cust_last IS NULL THEN
                    RAISE EXCEPTION 200;
          END IF;

END IF;

SELECT o_id, o_entry_d, o_carrier_id, o_ol_cnt
          INTO order_id, entry_d, carrier_id, ol_cnt
          FROM order
          WHERE o_w_id = warehouse_id AND
               o_d_id = district_id AND
               o_c_id = customer_id AND
               o_id = (SELECT MAX(o_id) FROM order
                              WHERE o_w_id = warehouse_id
AND
                                   o_d_id = district_id AND
                                   o_c_id = customer_id );

LET i = 1;

FOREACH cur4 FOR SELECT ol_i_id, ol_supply_w_id,
ol_quantity, ol_amount,
          ol_delivery_d
          INTO i_id, supply_w_id, quantity, amount, delivery_d
          FROM order_line
          WHERE ol_w_id = warehouse_id
          AND   ol_d_id = district_id
          AND   ol_o_id = order_id

IF i < 9 THEN                    — 1 2 3 4 5 6 7 8
   IF i < 5 THEN                 — 1 2 3 4
      IF i < 3 THEN              — 1 2
         IF i < 2 THEN           — 1
                    LET i_id1,
supply_w_id1,quantity1,amount1, delivery_d1 =
                                   i_id,
supply_w_id,quantity,amount, delivery_d;
         ELSE                    — 2
                    LET i_id2,
supply_w_id2,quantity2,amount2, delivery_d2 =
                                   i_id,
supply_w_id,quantity,amount, delivery_d;
         END IF;
      ELSE                       — 3 4
         IF i < 4 THEN           — 3
                    LET i_id3,
supply_w_id3,quantity3,amount3, delivery_d3 =
                                   i_id,
supply_w_id,quantity,amount, delivery_d;
         ELSE                    — 4
                    LET i_id4,
supply_w_id4,quantity4,amount4, delivery_d4 =
                                   i_id,
supply_w_id,quantity,amount, delivery_d;
         END IF;
      END IF;
   ELSE                          — 5 6 7 8
      IF i < 7 THEN              — 5 6
         IF i < 6 THEN           — 5
                    LET i_id5,
supply_w_id5,quantity5,amount5, delivery_d5 =
                                   i_id,
supply_w_id,quantity,amount, delivery_d;
```

```
        ELSE                    — 6
                        LET i_id6,
supply_w_id6,quantity6,amount6, delivery_d6 =
                                i_id,
supply_w_id,quantity,amount, delivery_d;
        END IF;
      ELSE                      — 7 8
        IF i < 8 THEN           — 7
                        LET i_id7,
supply_w_id7,quantity7,amount7, delivery_d7 =
                                i_id,
supply_w_id,quantity,amount, delivery_d;
        ELSE                    — 8
                        LET i_id8,
supply_w_id8,quantity8,amount8, delivery_d8 =
                                i_id,
supply_w_id,quantity,amount, delivery_d;
        END IF;
      END IF;
  END IF;
ELSE                            — 9 10 11 12 13 14 15
   IF i < 13 THEN               — 9 10 11 12
      IF i < 11 THEN            — 9 10
        IF i < 10 THEN          — 9
                        LET i_id9,
supply_w_id9,quantity9,amount9, delivery_d9 =
                                i_id,
supply_w_id,quantity,amount, delivery_d;
        ELSE                    — 10
                        LET i_id10,
supply_w_id10,quantity10,amount10, delivery_d10 =
                                i_id,
supply_w_id,quantity,amount, delivery_d;
        END IF;
          ELSE                    — 11 12
        IF i < 12 THEN          — 11
                        LET  i_id11,
supply_w_id11,quantity11,amount11, delivery_d11 =
                               i_id,
supply_w_id,quantity,amount, delivery_d;
        ELSE                    — 12
                        LET i_id12,
supply_w_id12,quantity12,amount12, delivery_d12 =
                                i_id,
supply_w_id,quantity,amount, delivery_d;
        END IF;
      END IF;
   ELSE                         — 13 14 15
      IF i < 15 THEN            — 13 14
        IF i < 14 THEN          — 13
                        LET i_id13,
supply_w_id13,quantity13,amount13, delivery_d13 =
                                i_id,
supply_w_id,quantity,amount, delivery_d;
        ELSE                    — 14
                        LET i_id14,
supply_w_id14,quantity14,amount14, delivery_d14 =
                                i_id,
supply_w_id,quantity,amount, delivery_d;
        END IF;
      ELSE                      — 15
                        LET i_id15,
supply_w_id15,quantity15,amount15, delivery_d15 =
                                i_id,
supply_w_id,quantity,amount, delivery_d;
      END IF;
   END IF;
END IF;

— IF i = ol_cnt THEN
—         EXIT FOREACH;
— END IF;

LET i = i + 1;

END FOREACH;

COMMIT WORK;

RETURN 0, 0, customer_id, cust_first, cust_middle, cust_last,
          cust_balance, order_id, entry_d, carrier_id, ol_cnt,
          i_id1, supply_w_id1, quantity1, amount1,
delivery_d1,
          i_id2, supply_w_id2, quantity2, amount2,
delivery_d2,
          i_id3, supply_w_id3, quantity3, amount3,
delivery_d3,
          i_id4, supply_w_id4, quantity4, amount4,
delivery_d4,
          i_id5, supply_w_id5, quantity5, amount5,
delivery_d5,
          i_id6, supply_w_id6, quantity6, amount6,
delivery_d6,
          i_id7, supply_w_id7, quantity7, amount7,
delivery_d7,
          i_id8, supply_w_id8, quantity8, amount8,
delivery_d8,
          i_id9, supply_w_id9, quantity9, amount9,
delivery_d9,
          i_id10, supply_w_id10, quantity10, amount10,
delivery_d10,
          i_id11, supply_w_id11, quantity11, amount11,
delivery_d11,
          i_id12, supply_w_id12, quantity12, amount12,
delivery_d12,
          i_id13, supply_w_id13, quantity13, amount13,
delivery_d13,
          i_id14, supply_w_id14, quantity14, amount14,
delivery_d14,
          i_id15, supply_w_id15, quantity15, amount15,
delivery_d15
;

END PROCEDURE;

COMMIT WORK;
```

delivery.sql

```
SET OPTIMIZATION HIGH;

BEGIN WORK;

DROP PROCEDURE delivery;

CREATE PROCEDURE delivery (
w_id INT,
carrier_id SMALLINT,
p_ol_delivery_d DATETIME YEAR TO SECOND
```

```
)
RETURNING INT, INT, INT, INT, INT,
          INT, INT, INT, INT, INT,
          INT,        — sql_err
          INT;        — isam_err

DEFINE ol_total  INT;
DEFINE sql_err, isam_err INT;

DEFINE d_id SMALLINT;
DEFINE  cid,
        no_oid,
        o_id_1,
        o_id_2,
        o_id_3,
        o_id_4,
        o_id_5,
        o_id_6,
        o_id_7,
        o_id_8,
        o_id_9,
        o_id_10 INTEGER;

— SET DEBUG FILE TO '/tmp/delivery.trc';
— TRACE ON;
— SET EXPLAIN ON;

LET     d_id,
        o_id_1,
        o_id_2,
        o_id_3,
        o_id_4,
        o_id_5,
        o_id_6,
        o_id_7,
        o_id_8,
        o_id_9,
        o_id_10,
        sql_err,
        isam_err =
        1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;

WHILE d_id <= 10

BEGIN
   ON EXCEPTION SET sql_err, isam_err
      IF isam_err = -143
      THEN
         ROLLBACK WORK;
         LET sql_err = 0;
         WHILE isam_err < 0
            LET isam_err = isam_err + 1;
         END WHILE;
         CONTINUE WHILE;
      ELSE
         ROLLBACK WORK;
         EXIT WHILE;
       END IF;
   END EXCEPTION;

   SELECT MIN(no_o_id)
      INTO no_oid
   FROM new_order
   WHERE no_d_id = d_id AND no_w_id = w_id;

   IF no_oid IS NULL
   THEN
      LET no_oid = -100;

   ELSE
      DELETE FROM new_order WHERE no_o_id = no_oid
                AND no_d_id = d_id
                AND no_w_id = w_id;

      FOREACH order_cur FOR
         SELECT o_c_id
            INTO cid
            FROM order
            WHERE o_id = no_oid AND o_d_id = d_id AND
o_w_id = w_id

         UPDATE order SET o_carrier_id = carrier_id
            WHERE CURRENT OF order_cur;

        EXIT FOREACH;
      END FOREACH;

      UPDATE order_line SET ol_delivery_d = p_ol_delivery_d
         WHERE ol_o_id = no_oid AND ol_d_id = d_id AND
ol_w_id = w_id;

      SELECT SUM(ol_amount)
         INTO ol_total
         FROM order_line
         WHERE ol_o_id = no_oid 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 + 1
            WHERE c_id = cid AND c_d_id = d_id AND c_w_id
= w_id;
   END IF;

   COMMIT WORK;


   IF d_id < 6
   THEN
      IF d_id < 3
      THEN
         IF d_id < 2
         THEN
            LET o_id_1 = no_oid;
         ELSE
            LET o_id_2 = no_oid;
              END IF;
      ELSE
         IF d_id < 4
         THEN
            LET o_id_3 = no_oid;
         ELSE
                     IF d_id < 5
                     THEN
               LET o_id_4 = no_oid;
            ELSE
               LET o_id_5 = no_oid;
                 END IF;
             END IF;
      END IF;
   ELSE
      IF d_id < 8
      THEN
         IF d_id < 7
```

```
         THEN
            LET o_id_6 = no_oid;
         ELSE
            LET o_id_7 = no_oid;
              END IF;
      ELSE
         IF d_id < 9
         THEN
            LET o_id_8 = no_oid;
         ELSE
                     IF d_id < 10
                     THEN
               LET o_id_9 = no_oid;
            ELSE
               LET o_id_10 = no_oid;
               EXIT WHILE;
            END IF;
        END IF;
     END IF;
  END IF;

  LET d_id = d_id + 1;

END;

END WHILE

RETURN  o_id_1, o_id_2, o_id_3, o_id_4, o_id_5,
          o_id_6, o_id_7, o_id_8, o_id_9, o_id_10,
          sql_err, isam_err;

END PROCEDURE;

COMMIT WORK;
```

Appendix B - Database De

crtedb.ec

```
/*****************************************************************************/
/*      file:  crtedb.ec                                                     */
/*                                                                           */
/*****************************************************************************/

$include sqlca ;

/*---------------------------------------------------------------------------*/
/*      main                                                                 */
/*---------------------------------------------------------------------------*/


main ()
{
char str[80];

printf("This program destroys, and re-creates TPC-C database\n");
printf("Do you want to do this ? (y or n) \n");
scanf("%s", str);
if (str[0] == 'n') exit(1);

$ drop database tpcc ;
if (sqlca.sqlcode != 0) SQLWARN(__LINE__);

$ create database tpcc;
if (sqlca.sqlcode != 0) SQLERR(__LINE__);

printf("\nProgram finished successfully.\n");
}

/*---------------------------------------------------------------------------*/
/*      SQLERR                                                               */
/*---------------------------------------------------------------------------*/
SQLERR(at)
        int     at;
{
printf ("ERROR AT %d: SQLCA.SQLCODE=%d, ISAM Code=%d \n",
```

```
        at, sqlca.sqlcode,sqlca.sqlerrd[1]);
exit(-1);
}

/*------------------------------------------------------------
------------*/
/*      SQLWARN                                               */
/*------------------------------------------------------------
------------*/
SQLWARN(at)
        int     at;
{
printf ("WARNING AT %d: SQLCODE=%d, ISAM Code=%d \n",
        at, sqlca.sqlcode,sqlca.sqlerrd[1]);
}
```

crtetbl.ec

```
/************************************************************************/
/*      file:  load.ec                                                  */
/*                                                                      */
/*    TPC-C benchmark of Informix. this program creates the             */
/*    required tables, creates the randomly generated data,             */
/*     and loads the data into the tables.                              */
/*                                                                      */
/************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include "../Incl/appl.h"
#include "../Incl/create.h"
#define SQLERR()   if (sqlca.sqlcode != 0) sqlerr(__LINE__)
#define SQLWARN()          if (sqlca.sqlcode != 0)
sqlwarn(__LINE__)
void                    sqlerr(long at);
void                    sqlwarn(long at);
/*------------------------------------------------------------
------------*/
/*      main                                                    */
/*------------------------------------------------------------
------------*/
main ()
{
  long                  option;
  char                  str[80];
  char                  hashidx[128];
  char                  rows_per_page[32];
  $ char    idx_st[256];
  $ char    statement[512];
 printf("\n");
 printf("This program destroys and re-creates TPC-C tables\n");
 printf("Do you want to do this ? (y or n) \n");
 scanf("%s", str);
 if (str[0] == 'n')
 {
  EXIT(1);
 }
 printf("\n\n");
 printf("Pick table to be created: \n");
 printf("1. warehouse \n");
 printf("2. district \n");
 printf("3. item \n");
 printf("4. stock \n");
 printf("5. customer \n");
 printf("6. history \n");
 printf("7. orders \n");
 printf("8. order_line \n");
 printf("9. new_order \n");
 printf("\nEnter option: \n");
 scanf("%d", &option);
 if (option < 1 || option > 9)
 {
  printf("option %d illegal\n", option);
  EXIT(1);
 }
 option--;
 sprintf(hashidx, "HASHIDX=%s", hashidx_tbl[option]);
      sprintf(rows_per_page, "ROWS_PER_PAGE=%d",
rows_per_page_tbl[option]);
 putenv(hashidx);
 putenv(rows_per_page);
  $ database tpcc;
 SQLERR();
 printf("building table %s ...\n", tablename[option]);
 sprintf(statement, "drop table %s", tablename[option]);
 $ prepare drop_table from :statement;
 SQLWARN();
 if (sqlca.sqlcode == 0)
 {
  $ execute drop_table;
  SQLERR();
 }
 sprintf(statement, "%s in %s extent size %d next size %d lock
mode %s",
        createtbl[option], dbspace[option], extent_size[option],
        nextent_size[option], lockmode[option]);
 $ prepare create_table from :statement;
 SQLERR();
 $ execute create_table;
 SQLERR();
 if (strcmp(index_before_load[option], "") != 0)
 {
  if (strcmp(index_dbs[option], "") != 0)
      sprintf(statement, "%s in %s", index_before_load[option],
index_dbs[option]);
   else
    sprintf(statement, "%s", index_before_load[option]);
  printf("%s\n", statement);
  $ prepare create_idx1 from :statement;
  SQLERR();
  $ execute create_idx1;
  SQLERR();
 }
 if (strcmp(index_hash[option], "") != 0)
 {
  if (strcmp(index_dbs[option], "") == 0)
   {
    printf("no dbspace for detached index specified!\n");
     exit(1);
   }
  sprintf(idx_st, index_hash[option], "unique");
/*
   sprintf(statement, "%s in %s", idx_st, index_dbs[option]);
*/
```

```
    sprintf(statement, "%s in rootdbs", idx_st);
    printf("%s\n", statement);
    $ prepare create_idx2 from :statement;
    SQLERR();
    $ execute create_idx2;
    SQLERR();
    sprintf(statement, index_hash[option], "hash");
    printf("%s\n", statement);
    $ prepare create_hash from :statement;
    SQLERR();
    $ execute create_hash;
    SQLERR();
  }
  sprintf(statement, "grant all on %s to public", tablename[option]);
  $ prepare grant_table from :statement;
  SQLERR();
  $ execute grant_table;
  SQLERR();
  $ grant dba to public;
  SQLERR();
  $ close database;
  SQLERR();
  return(0);
}
```

create.h

```
/*************************************************************************/
/*                                                                       */
/*      File: create.h  */
/*                                                                       */
/*    This module contains global variables and data definitions    */
/*      for the creation of a tpcc database.
        */
/*                                                                       */
/*************************************************************************/
#define SQLERR()    if (sqlca.sqlcode != 0) sqlerr(__LINE__)
#define SQLWARN()           if      (sqlca.sqlcode      !=      0)
sqlwarn(__LINE__)
#define C_C_LAST   173
#define C_OL_I_ID   1723
#define A_C_LAST   255
#define NO_LOW_O_ID          2101
#define CUSTOMERS_PER_DISTRICT     3000     /* value = 3,000
*/
#define ITEMS                                 100000              /    *
value = 100,000 */
#define ORDER_LINES_PER_ORDER      10                    /    *
value >= 5 & <= 15 */
#define NU_ORDERS_PER_DISTRICT     900       /* value >= 900
*/
#define RandVal        lrand48
#define RandSeed       srand48
char         tablename[9][16] = {
             "warehouse",
             "district",
             "item",
             "stock",
             "customer",
             "history",
             "orders",
             "order_line",
             "new_order"
};
char         dbspace[9][16] = {
             "wdihno_tbl",          /* warehouse */
             "wdihno_tbl",          /* district */
             "wdihno_tbl",          /* item */
             "s_tbl",     /* stock */
             "c_tbl",     /* customer */
             "wdihno_tbl",          /* history */
             "o_tbl",     /* orders */
             "ol_tbl",    /* order_line */
             "wdihno_tbl"           /* new_order */
};
char         index_dbs[9][16] = {
             "rootdbs", /* warehouse */
             "rootdbs", /* district */
             "rootdbs", /* item */
             "rootdbs", /* stock */
             "c_idx_tbl",           /* customer */
             "",                    /* history */
             "o_idx_tbl",           /* orders */
             "rootdbs", /* order_line */
             ""                     /* new_order */
};
long         extent_size[9] = {
             256,                   /* warehouse */
             2048,                  /* district */
             10000,                 /* item */
             2067000,  /* stock */
             2067000,  /* customer */
             1625000,  /* history */
             913000,                /* orders */
             2095000,  /* order_line */
             400000                 /* neworder */
};
long         nextent_size[9] = {
             16,                    /* warehouse */
             32,                    /* district */
             1024,                  /* item */
             2068704,  /* stock */
             2068704,  /* customer */
             8192,                  /* history */
             1800480,  /* orders */
             2096688,  /* order_line */
             8192                   /* neworder */
};
char         lockmode[9][16] = {
             "row",                 /* warehouse */
             "row",                 /* district */
             "row",                 /* item */
             "page",                /* stock */
             "page",                /* customer */
             "row",                 /* history */
             "row",                 /* orders */
             "page",                /* order_line */
             "row"                  /* new_order */
};
short        fillfactor[9] = {
             100,                   /* warehouse */
```

```
        100,            /* district */
        100,            /* item */
        100,            /* stock */
        100,            /* customer */
        100,            /* history */
        100,            /* orders */
        100,            /* order_line */
        50              /* neworder */
};
char hashidx_tbl[9][64] = {
        "1:1:930:1:1:930",                      /* warehouse */
        "2:1:930:1:10:1:1:930",                 /* district */
        "1:1:100000:1:1:100000",                /* item */
        "2:1:100000:1:930:1:1:100000",          /* stock */
        "3:1:3000:1:930:1:10:1:1:3000",         /* customer */
        "",                                     /* history */
        "3:1:3200:1:930:1:10:1:1:3200",         /* orders */
        "4:1:3200:1:930:1:10:1:15:1:1:3200",    /* order_line */
        ""                                      /* new_order */
        };
short rows_per_page_tbl[9] = {
        10,     /* warehouse */
        10,     /* district */
        22,     /* item */
        6,      /* stock */
        3,      /* customer */
        38,     /* history */
        67,     /* orders */
        30,     /* order_line */
        168     /* new_order */
        };
char    createtbl[9][1024] = {
        /* Warehouse table */
        "create table warehouse (w_id smallint not null, "
                "w_name char(10) not null, "
                "w_street_1 char(20) not null, "
                "w_street_2 char(20) not null, "
                "w_city char(20) not null, "
                "w_state char(2) not null, "
                "w_zip char(9) not null, "
                "w_tax decimal(4,4) not null, "
                "w_ytd decimal(12,2) not null) ",
        /* District table */
        "create table district (d_id smallint not null, "
                "d_w_id smallint not null, "
                "d_name char(10) not null, "
                "d_street_1 char(20) not null, "
                "d_street_2 char(20) not null, "
                "d_city char(20) not null, "
                "d_state char(2) not null, "
                "d_zip char(9) not null, "
                "d_tax decimal(4,4) not null, "
                "d_ytd decimal(12,2) not null, "
                "d_next_o_id integer not null) ",
        /* Item table */
        "create table item (i_id integer not null, "
                "i_im_id integer not null, "
                "i_name char(24) not null, "
                "i_price decimal(5,2) not null, "
                "i_data char(50) not null) ",
        /* Stock table */
        "create table stock (s_i_id integer not null, "
                "s_w_id smallint not null, "
                "s_quantity smallint not null, "
                "s_dist_01 char(24) not null, "
                "s_dist_02 char(24) not null, "
                "s_dist_03 char(24) not null, "
                "s_dist_04 char(24) not null, "
                "s_dist_05 char(24) not null, "
                "s_dist_06 char(24) not null, "
                "s_dist_07 char(24) not null, "
                "s_dist_08 char(24) not null, "
                "s_dist_09 char(24) not null, "
                "s_dist_10 char(24) not null, "
                "s_ytd integer not null, "
                "s_order_cnt smallint not null, "
                "s_remote_cnt smallint not null, "
                "s_data char(50) not null) ",
        /* Customer table */
        "create table customer (c_id integer not null, "
                "c_d_id smallint not null, "
                "c_w_id smallint not null, "
                "c_first char(16) not null,
```

```
“
                                “c_middle       char(2) not null,
“
                                “c_last         char(16) not null,
“
                               “c_street_1     char(20) not null,
“
                               “c_street_2     char(20) not null,
“
                                “c_city         char(20) not null,
“
                                “c_state        char(2) not null,
“
                                “c_zip          char(9) not null, “
                               “c_phone        char(16) not null,
“
                                “c_since        datetime year to
second not null, “
                               “c_credit       char(2) not null, “
                                “c_credit_lim    decimal(12,2)
not null, “
                               “c_discount     decimal(4,4) not
null, “
                                “c_balance      decimal(12,2)
not null, “
                                “c_ytd_payment
decimal(12,2) not null, “
                                “c_payment_cnt
smallint not null, “
                                  “c_delivery_cnt  smallint not
null, “
                               “c_data         char(500) not null)
“,
        /* History table */
        “create table history (h_c_id   integer not null, “
                               “h_c_d_id smallint not null, “
                               “h_c_w_id smallint not null, “
                               “h_d_id   smallint not null, “
                               “h_w_id   smallint not null, “
                               “h_date   datetime year to sec-
ond not null, “
                              “h_amount decimal(6,2) not null,
“
                               “h_data   char(24) not null) “,
        /* Orders table */
        “create table orders (o_id        integer not null, “
                               “o_c_id          integer not null,
“
                               “o_d_id          smallint not
null, “
                               “o_w_id          smallint not
null, “
                               “o_entry_d       datetime year
to second not null, “
                               “o_carrier_id    smallint not
null, “
                               “o_ol_cnt        smallint not
null, “
                               “o_all_local     smallint not
null) “,
        /* Order_line table */
        “create table order_line (ol_o_id          integer not null,
“
                               “ol_d_id smallint not
null, “
                               “ol_w_id smallint not
null, “
                               “ol_number
smallint not null, “
                               “ol_i_id   integer not null,
“
                               “ol_supply_w_id
smallint not null, “
                               “ol_delivery_d
datetime year to second, “
                               “ol_quantity
smallint not null, “
                               “ol_amount
decimal(6,2) not null, “
                               “ol_dist_info
char(24) not null) “,
        /* New_order table */
        “create table new_order (no_o_id          integer not null,
“
                                “no_d_id        smallint not
null, “
                                “no_w_id        smallint not
null) “
};
char    index_before_load[9][128] = {
        “”,     /* warehouse */
        “”,     /* district */
        “”,     /* item */
        “”,     /* stock */
        “”,     /* customer */
        “”,     /* history */
                /* orders */
        “create unique index o_idx1 on orders(o_c_id, o_w_id,
o_d_id, o_id desc) “,
        “”,     /* order_line */
                /* new_order */
        “”
};
char    index_hash[9][128] = {
        /* Warehouse index */
        “create %s index w_idx1 on warehouse(w_id) “,
        /* District index */
        “create %s index d_idx1 on district(d_w_id, d_id) “,
        /* Item index */
        “create %s index i_idx1 on item(i_id) “,
        /* Stock index */
        “create %s index s_idx1 on stock(s_i_id, s_w_id) “,
        /* Customer index */
        “create %s index c_idx1 on customer(c_id, c_w_id,
c_d_id) “,
        /* History index */
        “”,
        /* Orders index */
        “create %s index o_idx2 on orders(o_id, o_w_id, o_d_id)
“,
        /* Order_line index */
        “create %s index ol_idx1 on order_line(ol_o_id, ol_w_id,
ol_d_id, ol_number) “,
        /* New_order index */
        “”
};
```

```
char        index_after_load[9][128] = {
            "",         /* warehouse */
            "",         /* district */
            "",         /* item */
            "",         /* stock */
                        /* customer */
            "create index c_idx2 on customer(c_last, c_w_id, c_d_id,
c_first) ",
            "",         /* history */
            "",         /* orders */
            "",         /* order_line */
                        /* new_order */
            "create unique index no_idx1 on new_order(no_w_id,
no_o_id, no_d_id) "
};
/*————————————————————————————————————————————————————————————
————————————*/
/*      sqlerr                                                */
/*————————————————————————————————————————————————————————————
————————————*/
void sqlerr(at)
long        at;
{
 printf ("ERROR AT %d: SQLCA.SQLCODE=%d, ISAM Code=%d
\n",
        at, sqlca.sqlcode, sqlca.sqlerrd[1]);
 EXIT(-1);
}
/*————————————————————————————————————————————————————————————
————————————*/
/*      sqlwarn                                               */
/*————————————————————————————————————————————————————————————
————————————*/
void sqlwarn(at)
long        at;
{
  if (sqlca.sqlcode == -206)
  {
    printf("AT: %d, — no table to drop —\n", at);
    return;
  }
  printf("WARNING AT %d: SQLCODE=%d, ISAM Code=%d \n",
        at, sqlca.sqlcode, sqlca.sqlerrd[1]);
}
```

load.ec

```
/*********************************************************************/
/*      file:  load.ec                                              */
/*                                                                  */
/*      TPC-C benchmark of Informix. this program creates the
randomly */
/*          generated data, and loads the data into the tables.     */
/*                                                                  */
/*********************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include "../Incl/appl.h"
#include "../Incl/create.h"
/*——————————————*/
/*  PROTOTYPES.  */
/*——————————————*/
long            RandAstr(char *buffer, long x, long y);
long            RandNstr(char *buffer, long x, long y);
static long     NURand (long A, long x, long y, long C);
static long     RandInt (long x, long y );
float           RandFloat(long x, long y, long precision);
long            RandAstrOrg(char *buffer, long x, long y, long
percent);
long            RandLastName(char *buffer, long c_id);
long            RandZip(char *buffer);
void            sqlerr(long at);
void            sqlwarn(long at);
/*————————————————————*/
/* global variables */
/*————————————————————*/
char      CustPermTab[CUSTOMERS_PER_DISTRICT];
char      *syllables[] = { "BAR", "OUGHT", "ABLE", "PRI", "PRES",
              "ESE", "ANTI", "CALLY", "ATION", "EING"};
long      ware_count;
long      option;
/*————————————————————————————————————————————————————————————
——————————————*/
/*      main                                                  */
/*————————————————————————————————————————————————————————————
——————————————*/
main ()
{
  char              str[80];
  $ char    statement[512];
 printf("\n");
 printf("This program destroys, re-creates and loads TPC-C files\n");
  printf("Do you want to do this ? (y or n) \n");
  scanf("%s", str);
  if (str[0] == 'n')
  {
   EXIT(1);
  }
 printf("\n\n");
 printf("Enter number of warehouses —>");
 scanf("%d",&ware_count);
 printf("\n\n");
  printf("Pick table to be created: \n");
 printf("1. warehouse \n");
  printf("2. district \n");
  printf("3. item \n");
  printf("4. stock \n");
  printf("5. customer \n");
```

```
 printf("6. history \n");
 printf("7. orders (orders, order-line) \n");
 printf("9. new_order \n");
 printf("\nEnter option: \n");
 scanf("%d", &option);
 RandSeed(7);
 $ database tpcc;
 SQLERR();
 option—;
 if (option == 6) /* orders and order_line */
   printf("loading table %s and table %s ...\n", tablename[option],
tablename[option+1]);
 else
   printf("loading table %s ...\n", tablename[option]);
   sprintf(statement, "lock table %s in exclusive mode",
tablename[option]);
 printf("%s\n", statement);
 $ prepare lock_table1 from :statement;
 SQLERR();
 $ execute lock_table1;
 SQLERR();
 if (option == 6) /* orders, needs also to lock table order_line */
 {
      sprintf(statement, "lock table %s in exclusive mode",
tablename[option+1]);
   printf("%s\n", statement);
   $ prepare lock_table2 from :statement;
   SQLERR();
   $ execute lock_table2;
   SQLERR();
 }
 switch (option)
 {
            case 0: /* WAREHOUSE */
                        create_ware_tbl();
                        break;
            case 1: /* DISTRICT  */
                        create_dist_tbl();
                        break;
            case 2: /* ITEM      */
                        create_item_tbl();
                        break;
            case 3: /* STOCK     */
                        create_stock_tbl();
                        break;
            case 4: /* CUSTOMER  */
                        create_cust_tbl();
                        break;
            case 5: /* HISTORY   */
                        create_hist_tbl();
                        break;
            case 6: /* ORDERS    */
                        create_ordr_tbl();
                        break;
            case 8: /* NEW_ORDER */
                        create_nu_ord_tbl();
                        break;
            default:
                        printf("error: invalid option. \n");
                        break;
 }
 if (strcmp(index_after_load[option], "") != 0)
 {
    if (strcmp(index_dbs[option], "") != 0)
      sprintf(statement, "%s fillfactor %d in %s",
                         index_after_load[option], fillfactor[option],
index_dbs[option]);
    else
      sprintf(statement, "%s fillfactor %d",
            index_after_load[option], fillfactor[option]);
    printf("%s\n", statement);
    $ prepare create_idx from :statement;
    SQLERR();
    $ execute create_idx;
    SQLERR();
 }

   sprintf(statement, "update statistics for table %s",
tablename[option]);
 printf("%s\n", statement);
 $ prepare update_stat1 from :statement;
 SQLERR();
 $ execute update_stat1;
 SQLERR();
  if (option == 6) /* orders, needs also to update statistics for
order_line */
 {
      sprintf(statement, "update statistics for table %s",
tablename[option+1]);
    printf("%s\n", statement);
    $ prepare update_stat2 from :statement;
    SQLERR();
    $ execute update_stat2;
    SQLERR();
 }
 $ grant dba to public;
 SQLERR();
 $ close database;
 SQLERR();
 return(0);
}
/*————————————————————————————————————
——————*/
/*       create item table                                      */
/*————————————————————————————————————
——————*/
create_item_tbl()
{
  $ long    i_id;
  $ long    i_im_id;
  $ char    i_name[25];
  $ float   i_price;
  $ char    i_data[51];
  $ declare item_ins cursor with hold for
   insert into item values(:i_id, :i_im_id, :i_name, :i_price, :i_data);
 SQLERR();
 $ open item_ins;
 SQLERR();
 for (i_id = 1; i_id <= ITEMS; i_id++)
 {
   i_im_id = RandInt(1, 10000);                     /* create image
field */
   RandAstr(i_name, 14, 24);                        /* create name
field */
   i_price = RandFloat(100, 10000, 100);/* create price field */
   RandAstrOrg(i_data, 26, 50, 10);                 /* create data
```

```
field */
    $ put item_ins;
    SQLERR();
    if ((i_id % 5000) == 0)
      printf("loaded item %d\n", i_id);
  }
  $ close item_ins;
  SQLERR();
}
/*---------------------------------------------------------------*/
/*      create stock table                                       */
/*---------------------------------------------------------------*/
create_stock_tbl()
{
  $ long    s_i_id;
  $ long    s_w_id;
  $ long    s_quantity;
  $ char    s_dist_01[25];
  $ char    s_dist_02[25];
  $ char    s_dist_03[25];
  $ char    s_dist_04[25];
  $ char    s_dist_05[25];
  $ char    s_dist_06[25];
  $ char    s_dist_07[25];
  $ char    s_dist_08[25];
  $ char    s_dist_09[25];
  $ char    s_dist_10[25];
  $ char    s_data[51];
  $ declare stock_ins cursor with hold for
    insert into stock values(:s_i_id, :s_w_id, :s_quantity,:s_dist_01,
                             :s_dist_02, :s_dist_03, :s_dist_04, :s_dist_05,
                             :s_dist_06, :s_dist_07, :s_dist_08, :s_dist_09,
                              :s_dist_10, 0, 0, 0, :s_data);
  SQLERR();
  $ open stock_ins;
  SQLERR();
  for (s_i_id = 1; s_i_id <= ITEMS; s_i_id++)
  {
    for (s_w_id = 1; s_w_id <= ware_count; s_w_id++)
    {
      s_quantity = RandInt(10, 100);
      RandAstr(s_dist_01, 24, 24);          /* create text field */
      RandAstr(s_dist_02, 24, 24);          /* create text field */
      RandAstr(s_dist_03, 24, 24);          /* create text field */
      RandAstr(s_dist_04, 24, 24);          /* create text field */
      RandAstr(s_dist_05, 24, 24);          /* create text field */
      RandAstr(s_dist_06, 24, 24);          /* create text field */
      RandAstr(s_dist_07, 24, 24);          /* create text field */
      RandAstr(s_dist_08, 24, 24);          /* create text field */
      RandAstr(s_dist_09, 24, 24);          /* create text field */
      RandAstr(s_dist_10, 24, 24);          /* create text field */
      RandAstrOrg(s_data, 26, 50, 10);    /* create ORIGINAL field */
       $ put stock_ins;
      SQLERR();
      if ((s_w_id == ware_count) && ((s_i_id % 10) == 0))
        printf("loaded stock for item %d\n", s_i_id);
    }
  }
  $ close stock_ins;
  SQLERR();
}
/*---------------------------------------------------------------*/
/*      create warehouse table                                   */
/*---------------------------------------------------------------*/
create_ware_tbl()
{
  $ long    w_id;
  $ char    w_name[11];
  $ char    w_street_1[21];
  $ char    w_street_2[21];
  $ char    w_city[21];
  $ char    w_state[3];
  $ char    w_zip[10];
  $ float   w_tax;
  $ declare warehouse_ins cursor with hold for
      insert into warehouse values(:w_id, :w_name, :w_street_1,
:w_street_2,
                                   :w_city, :w_state, :w_zip, :w_tax,
300000.0 );
  SQLERR();
  $ open warehouse_ins;
  SQLERR();
  for (w_id = 1; w_id <= ware_count; w_id++)
  {
    RandAstr(w_name, 6, 10);              /* create name */
    RandAstr(w_street_1, 10, 20);         /* create street 1 */
    RandAstr(w_street_2, 10, 20);         /* create street 2 */
     RandAstr(w_city, 10, 20);            /* create city */
     RandAstr(w_state, 2, 2);             /* create state */
    RandZip(w_zip);                       /* create zip */
    w_tax = RandFloat(0, 2000, 10000);    /* create tax rate */
    $ put warehouse_ins;
    SQLERR();
    if ((w_id % 10) == 0)
      printf( "loaded Warehouse #%d\n", w_id);
  }
  $ close warehouse_ins;
  SQLERR();
}
/*---------------------------------------------------------------*/
/*      create dist table                                        */
/*---------------------------------------------------------------*/
create_dist_tbl()
{
  $ long    d_id;
  $ long    d_w_id;
  $ char    d_name[11];
  $ char    d_street_1[21];
  $ char    d_street_2[21];
  $ char    d_city[21];
  $ char    d_state[3];
  $ char    d_zip[10];
  $ float   d_tax;
  $ declare district_ins cursor with hold for
    insert into district values(:d_id, :d_w_id, :d_name, :d_street_1,
                               :d_street_2, :d_city, :d_state, :d_zip,
                               :d_tax, 30000.0, 3001);
  SQLERR();
  $ open district_ins;
  SQLERR();
```

```
 for (d_w_id = 1; d_w_id <= ware_count; d_w_id++)
 {
  for (d_id = 1; d_id <= DISTRICTS_PER_WAREHOUSE; d_id++)
   {
    RandAstr(d_name, 6, 10);                    /* create name
*/
    RandAstr(d_street_1, 10, 20);               /* create street
1 */
    RandAstr(d_street_2, 10, 20);               /* create street
2 */
     RandAstr(d_city, 10, 20);          /* create city */
     RandAstr(d_state, 2, 2);           /* create state */
    RandZip(d_zip);                             /* create zip */
    d_tax = RandFloat(0, 2000, 10000); /* create tax rate */

     $ put district_ins;
     SQLERR();
   }
   if ((d_w_id % 10) == 0)
    printf("loaded Districts for Warehouse %d\n", d_w_id);
 }
 $ close district_ins;
 SQLERR();
}
/*-------------------------------------------------------------
-------------*/
/*      create customer table                                  */
/*-------------------------------------------------------------
-------------*/
create_cust_tbl()
{
 $ long    c_id;
 $ long    c_w_id;
 $ long    c_d_id;
 $ char    c_last[17];
 $ char    c_first[17];
 $ char    c_street_1[21];
 $ char    c_street_2[21];
 $ char    c_city[21];
 $ char    c_state[3];
 $ char    c_zip[10];
 $ char    c_phone[17];
 $ char    c_credit[3];
 $ char    c_data[501];
 $ float   c_discount;
 $ declare customer_ins cursor with hold for
   insert into CUSTOMER values(:c_id, :c_d_id, :c_w_id, :c_first,
"OE", :c_last,
                            :c_street_1, :c_street_2, :c_city, :c_state,
:c_zip,
                            :c_phone, CURRENT, :c_credit, 50000.0,
:c_discount,
                           -10.0, 10.0, 1, 0, :c_data );
 SQLERR();
 $ open customer_ins;
 SQLERR();
 for (c_id = 1; c_id <= CUSTOMERS_PER_DISTRICT; c_id++)
 {
  for (c_w_id = 1; c_w_id <= ware_count; c_w_id++)
   {
    for (c_d_id = 1; c_d_id <= DISTRICTS_PER_WAREHOUSE;
c_d_id++)
     {
      RandLastName(c_last, c_id);       /* create last name */
      if (RandInt(0, 1000)  < 100)
        strcpy( c_credit, "BC" );     /* create credit */
       else
        strcpy( c_credit, "GC" );     /* create credit */
       RandAstr(c_first, 8, 16);  /* create first name */
       RandAstr(c_street_1, 10, 20);       /* create street 1 */
       RandAstr(c_street_2, 10, 20);       /* create street 2 */
       RandAstr(c_city, 10, 20);           /* create city */
       RandAstr(c_state, 2, 2); /* create state */
       RandZip(c_zip);                     /* create zip */
       RandNstr(c_phone, 16, 16);          /* create phone number */
       RandAstr(c_data, 300,500);          /* create customer data */
       c_discount = RandFloat(0, 5000, 10000); /* create discount
rate */
       $ put customer_ins;
       SQLERR();
     }
    }
   printf("loaded customer c_id %d\n", c_id);
  }
  $ close customer_ins;
  SQLERR();
}
/*-------------------------------------------------------------
-------------*/
/*      create hist table                                      */
/*-------------------------------------------------------------
-------------*/
create_hist_tbl()
{
  $ long    h_c_id;
  $ long    h_d_id;
  $ long    h_w_id;
  $ char    h_data[25];
  $ declare history_ins cursor with hold for
    insert into history values(:h_c_id, :h_d_id, :h_w_id, :h_d_id,
                          :h_w_id, CURRENT, 10.0, :h_data);
  SQLERR();
  $ open history_ins;
  SQLERR();
  for (h_w_id = 1; h_w_id <= ware_count; h_w_id++)
  {
    for (h_d_id = 1; h_d_id <= DISTRICTS_PER_WAREHOUSE;
h_d_id++)
    {
      for (h_c_id = 1; h_c_id <= CUSTOMERS_PER_DISTRICT;
h_c_id++)
     {
       RandAstr(h_data, 12, 24);          /* create history data */
       $ put history_ins;
       SQLERR();
     }
      printf("loaded History for District %d for Warehouse %d\n",
h_d_id, h_w_id);
    }
  }
  $ close history_ins;
  SQLERR();
}
/*-------------------------------------------------------------
-------------*/
/*      create nu_ord table                                    */
```

```
/*---------------------------------------------------------------*/
create_nu_ord_tbl()
{
  $ long    no_o_id;
  $ long    no_d_id;
  $ long    no_w_id;
  $ char    statement[512];
  $ declare neworder_ins cursor with hold for
   insert into new_order values(:no_o_id, :no_d_id, :no_w_id);
  $ open neworder_ins;
  SQLERR();
        for (no_o_id=NO_LOW_O_ID; no_o_id <=
CUSTOMERS_PER_DISTRICT; no_o_id++)
  {
   for (no_w_id = 1; no_w_id <= ware_count; no_w_id++)
    {
   for (no_d_id = 1; no_d_id <= DISTRICTS_PER_WAREHOUSE;
no_d_id++)
     {
           $ put neworder_ins;
      SQLERR();
     }
    }
   if ((no_o_id % 5) == 0)
    printf("neworder no_o_id %d completed\n", no_o_id);
  }
  $ close neworder_ins;
  SQLERR();
#ifdef NEVER
  sprintf(statement, "alter table new_order index only");
  $ prepare alter_neworder from :statement;
  SQLERR();
  $ execute alter_neworder;
  SQLERR();
#endif /* NEVER */
}
/*---------------------------------------------------------------*/
/*      create order and line items tables                       */
/*---------------------------------------------------------------*/
create_ordr_tbl()
{
  $ long    o_id;
  $ long    o_c_id;
  $ long    o_w_id;
  $ long    o_d_id;
  $ datetime year to second     o_entry_d;
  $ long    o_carrier_id;
  $ long    o_ol_cnt;
  $ long    ol_number;
  $ long    ol_i_id;
  $ float   ol_amount;
  $ char    ol_dist_info[25];
  $ declare order_ins cursor with hold for
   insert into orders values(:o_id, :o_c_id, :o_d_id, :o_w_id,
                             :o_entry_d, :o_carrier_id, :o_ol_cnt, 1);
  SQLERR();
  $ declare orderline_ins_1 cursor with hold for
   insert into order_line values(:o_id, :o_d_id, :o_w_id, :ol_number,
                                 :ol_i_id, :o_w_id, :o_entry_d, 5,
                                 :ol_amount, :ol_dist_info);
  SQLERR();
  $ declare orderline_ins_2 cursor with hold for
   insert into order_line values(:o_id, :o_d_id, :o_w_id, :ol_number,
                                 :ol_i_id, :o_w_id, NULL, 5,
                                 :ol_amount, :ol_dist_info);
  SQLERR();
  $ open order_ins;
  SQLERR();
  $ open orderline_ins_1;
  SQLERR();
  $ open orderline_ins_2;
  SQLERR();
  for (o_w_id = 1; o_w_id <= ware_count; o_w_id++)
  {
    for (o_d_id = 1; o_d_id <= DISTRICTS_PER_WAREHOUSE;
o_d_id++)
    {
     PermSeed();
    for (o_id = 1; o_id <= CUSTOMERS_PER_DISTRICT; o_id++)
     {
      dtcurrent(&o_entry_d);
      if (o_id < NO_LOW_O_ID)
       o_carrier_id = RandInt(1, 10);
      else
       o_carrier_id = -1;
           o_c_id = RandPerm();
          o_ol_cnt = RandInt(ORDER_LINES_PER_ORDER-5,
ORDER_LINES_PER_ORDER+5);
       $ put order_ins;
       SQLERR();
      for (ol_number = 1; ol_number <= o_ol_cnt; ol_number++ )
       {
        ol_i_id = RandInt(1, 100000);
        RandAstr(ol_dist_info, 24, 24);
             if (o_id < NO_LOW_O_ID)
        {
          ol_amount = 0.0;
          $ put  orderline_ins_1;
          SQLERR();
        }
        else
        {
         ol_amount = RandFloat(1, 999999, 100);
          $ put  orderline_ins_2;
          SQLERR();
        }
       }
      }
      printf("loaded Orders for District %d on Warehouse %d\n",
o_d_id, o_w_id);
    }
  }
  $ close order_ins;
  SQLERR();
  $ close orderline_ins_1;
  SQLERR();
  $ close orderline_ins_2;
  SQLERR();
}
/*---------------------------------------------------------------*/
/*   PermSeed
                */
```

```
/*---------------------------------------------------------------
---------------*/
PermSeed()
{
  static long           i;
 for (i=0; i<CUSTOMERS_PER_DISTRICT; i++)
    CustPermTab[i] = 0;
}
/*---------------------------------------------------------------
---------------*/
/*   RandPerm
                */
/*   random permutation for customer ids
*/
/*---------------------------------------------------------------
---------------*/
RandPerm()
{
  static long           i, x;
 x = RandInt(0, CUSTOMERS_PER_DISTRICT - 1);
 for (i=0; i<CUSTOMERS_PER_DISTRICT; i++)
  {
    if (CustPermTab[x] == 0)
    {
      CustPermTab[x] = 1;
      return(x+1);
    }
    else
      x++;
   if (x == CUSTOMERS_PER_DISTRICT)
      x = 0;
  }
 printf("\nfatal error in RandPerm \n");
 EXIT(1);
}
/*---------------------------------------------------------------
---------------*/
/*   RandAstr
                        */
/*
                        */
/*  create a random alphanumeric string, of random length between
x               */
/*   and y and place them in designated buffer. Routine returns the
*/
/*    actual length.
                */
/*
                        */
/*   parameters
                        */
/*   ----------
                        */
/*   x end of acceptable length range
                */
/*   y end of acceptable length range
                */
/*
                        */
/*   output
                */
/*   ------
                        */
/*   actual length
                */
/*   random alphanumeric string
                        */
/*---------------------------------------------------------------
---------------*/
long RandAstr(char *buffer, long x, long y)
{
  long i;
  long wdlngth;
  long zero = '0';
  long z = 'z';
  char word[512];
  wdlngth = RandInt(x, y);
  for (i=0; i<wdlngth; i++ )
  {
    do
    {
     word[i] = (char)RandInt(zero, z);
    } while (!((word[i] >= '0' && word[i] <= '9') ||
              (word[i] >= 'A' && word[i] <= 'Z') ||
              (word[i] >= 'a' && word[i] <= 'z')));
  }
  word[wdlngth] = '\0';
  strcpy(buffer, word);
 return(wdlngth);
}
/*---------------------------------------------------------------
---------------*/
/*   RandNstr
                        */
/*
                        */
/*  create a random numeric string, of random length between x and
*/
/*   y and place them in designated buffer. Routine returns the
*/
/*    actual length.
                */
/*
                        */
/*   parameters
                        */
/*   ----------
                        */
/*   x end of acceptable length range
                */
/*   y end of acceptable length range
                */
/*
                        */
/*   output
                */
/*   ------
                        */
/*   actual length
                */
/*   random numeric string
                */
/*---------------------------------------------------------------
---------------*/
long RandNstr(char *buffer, long x, long y)
{
```

```
  long i;
  long numlngth;
  long zero = '0';
  long nine = '9';
  char number[64];
  numlngth = RandInt(x, y);
  for (i=0; i<numlngth; i++ )
    number[i] = (char)RandInt(zero, nine);
  number[numlngth] = '\0';
 strcpy(buffer, number);
 return(numlngth);
}
/*_____
_____*/
/*  NURand
                    */
/*
                    */
/*  create a non-uniform random numeric value of type integer, of
*/
/*   random value between lo and hi. Number is NOT placed in
BUFFER, */
/*  and IS simply RETURNED.
                    */
/*
                    */
/*  Routine RETURNS the VALUE.
                    */
/*
                    */
/*  parameters
                    */
/*  _________
                    */
/*  x end of acceptable value range
                */
/*  y end of acceptable value range
                */
/*
                    */
/*   output
                */
/*  _____
                    */
/*  random integer value  RETURNED
                */
/*
                    */
/*_____
_____*/
static long NURand (long A, long x, long y, long C)
{
 return((((RandInt(0,A)|RandInt(x,y))+C)%(y-x+1))+x);
}
/*_____
_____*/
/*  RandInt
                    */
/*
                    */
/*  create a uniform random numeric value of type integer, of random
*/
/*  value between x and y. Number is NOT placed in BUFFER, and
IS          */
/*  simply RETURNED.
                    */
/*
                    */
/*  Routine RETURNS the VALUE.
                    */
/*
                    */
/*  parameters
                    */
/*  _________
                    */
/*  x end of acceptable value range
                */
/*  y end of acceptable value range
                */
/*
                    */
/*   output
                */
/*  _____
                    */
/*  random integer value  RETURNED
                */
/*
                    */
/*_____
_____*/
static long RandInt (long x, long y)
{
 return((RandVal()%(y-x+1))+x);
}
/*_____
_____*/
/*  RandFloat
                    */
/*
                    */
/*  create a random numeric value of type float, of random value
*/
/*   y. Number is NOT placed in BUFFER, and IS simply RE-
TURNED.            */
/*
                    */
/*  Routine RETURNS the VALUE.
                    */
/*
                    */
/*  parameters
                    */
/*  _________
                    */
/*  x end of acceptable value range
                */
/*  y end of acceptable value range
                */
/*  the desired numeric type value is to be (integer, floating point).
*/
/*
                    */
/*   output
                */
```

```
/*   ———
                    */
/*  random value  RETURNED
          */
/*————————————————————————————
——————*/
float RandFloat(long x, long y, long precision)
{
 return((float)RandInt(x, y)/precision);
}
/*————————————————————————————
——————*/
/*   RandAstrOrg
          */
/*
                    */
/*  create a random alphanumeric string, of random length between
x         */
/*   and y and place them in designated buffer. Routine returns the
*/
/*   actual length.
          */
/*
                    */
/*  the word "ORIGINAL" is placed at a random location in the buffer
at*/
/*   random, for a given percent of the records.
*/
/*
                    */
/*   percent_to_set must be a floating point value from 0.0 to 100.0
*/
/*    if 0.0, no records will be set. If 100.0, all records will be set.
*/
/*
                    */
/*   CANNOT USE ON STRINGS OF LENGTH LESS THAN 8 !
                    */
/*    LOWER LIMIT MUST BE > 8 !
                    */
/*
                    */
/*   parameters
                    */
/*   —————
                    */
/*   x end of acceptable length range
          */
/*   y end of acceptable length range
          */
/*   percentage of records to set to ORIGINAL
          */
/*
                    */
/*   output
          */
/*   ———
                    */
/*   actual length
          */
/*  random alphanumeric string with the word "ORIGINAL" is placed
at a      */
/*       random location
          */
/*————————————————————————————
——————*/
long RandAstrOrg(char *buffer, long x, long y, long percent)
{
 long wdlngth;
 long start_pos;
 wdlngth = RandAstr(buffer, x, y);
 if (RandInt(0, 100)  < percent)
 {
   start_pos = RandInt(0, wdlngth - 8);
  strncpy(buffer+start_pos, "ORIGINAL", 8);
 }
 return(wdlngth);
}
/*————————————————————————————
——————*/
/* RandLastName
                    */
/*
                    */
/* parameters:
                    */
/*    buffer - target buffer for the generated last name
*/
/*
                    */
/* description:
                    */
/*    RandLastName generates a random number from 0 to 999
          */
/*   inclusive. a random name is generated by associating a random
*/
/*    string with each digit of the generated number. the three strings
*/
/*    are concatenated to generate the name
          */
/*————————————————————————————
——————*/
long RandLastName(char *buffer, long c_id)
{
 long randnum;
 if (c_id <= 1000)
   randnum = c_id - 1;
  else
   randnum = NURand( A_C_LAST, 0, 999, C_C_LAST );
 strcpy(buffer, syllables[randnum / 100]);
 randnum %= 100;
 strcat(buffer, syllables[randnum / 10]);
 randnum %= 10;
 strcat(buffer, syllables[randnum]);
 return(strlen(buffer));
}
/*————————————————————————————
——————*/
/* RandZip
                    */
/*
                    */
/*  description:
          */
/*    generates a random zip code (4.3.2.7)
          */
```

```
/*_____
_____*/
long RandZip(char *buffer)
{
  RandNstr(buffer, 4, 4);
  memcpy(buffer + 4, “11111”, 5);
  buffer[9] = “\0”;

 return(strlen(buffer));
}
```

Appendix C - Tunable Parame

IRIX Parameters - SGI Origin2000

SERVER OS TUNABLE SETTINGS

```
showconfig                      = 1
nosuidshells                    = 0
* Informix TPC-C parameters
*maxdmasz                       = 0xc00000
maxdmasz                        = 0x100
maxlkmem              = 0x100000
rlimit_rss_max        = 0x70000000 ll
rlimit_rss_cur        = 0x70000000 ll
rlimit_nofile_cur     = 1024 ll
rsshogfrac                      = 98
syssegsz              = 1200000
nproc                 = 2100
maxup                 = 2000
semmni                = 2048
semmns                = 2048
semmnu                = 2048
semume                = 200
semmsl                = 400
semopm                = 400
shmmax                = 0x200000000  ll
shmmni                = 1000
* shmall               = 2048
sshmseg               = 1000
msgmnb                = 65536
msgmax                = 32768
msgmni                = 512
msgseg                = 8000
msgtql                = 256


nlpages_16m           = 340
nlpages_4m            = 462
nlpages_1m            = 26
nlpages_256k          = 8
nlpages_64k           = 58
*
percent_totalmem_16m_pages  = 50
```

DCS_INTR_MASK 0x3
LCE_INTR_MASK 0x4
NRAIOBUF 256

percent_totalmem_4m_pages = 16
percent_totalmem_1m_pages = 1
percent_totalmem_256k_pages = 1
percent_totalmem_64k_pages = 1
*

IRIX Parameters - Challenge S

OS TUNABLE SETTINGS

```
* This file contains local system settings for tunable parameters.
* The parameter settings in this file replace the default values
* specified in mtune, if the new values are within the legal range
* for the parameter specified in mtune. The file contains one line
* for each parameter to be reset. The syntax for each line is :
*
*       <parameter name> = <value>
*
*       parameter name: This is the external name of the tunable
*                       parameter used in the mtune files.
*       value: This field contains the new value for the tunable parameter.
*
* Here is an example of what the stune might look like:
*
* ngroups_max = 25

nproc   = 2000
msgseg  = 37684
msgtql  = 4000
msgssz  = 128
msgmni  = 1200
msgmnb  = 307200
msgmax  = 32768
strmsgsz = 32768
shmall  = 512
sshmseg = 100
shmmni  = 100
shmmin  = 1
shmmax  = 536870912
semaem  = 16384
semvmx  = 32767
semume  = 256
semopm  = 256
semmsl  = 256
semmnu  = 1536
semmns  = 1536
semmni  = 1536
sema_pool_size  = 8192
maxup   = 2000
gpgslo  = 2000
```

BEA Tuxedo 6.1 CFS Parameters

```
#
#————————————————————————————————#
*RESOURCES                                  #
#————————————————————————————————#
#
IPCKEY          70952           # IPC KEY from 32,768 to 16,777,215
UID             3596            # user id as displayed by command "id"
GID             994             # group id as displayed by command "id"
PERM            0666            # UNIX permission from 0001 to 0777 in octal
MAXACCESSERS    1000            # max no of processes accesing bulleting board
MAXSERVERS      200             # maximum number of servers
MAXSERVICES     200             # maximum number of services
MASTER          SITE1           # machine on which master copy is found
MODEL           SHM             # SHM=single processor, MP=multi processor
LDBAL           Y               # load balancing, Y=yes, N=no
MAXGTT          10              # maximum simultaneous global transactions
#MAXBUFTYPE     8               # maximum buffer types
#MAXBUFSTYPE    16              # maximum buffer subtypes
SCANUNIT        60              # scan program wake-up time in secs.
SANITYSCAN      5               # sanity scan wake-up
BBLQUERY        30              # check out wake-up time
BLOCKTIME       5               # blocking call time-out
TAGENT          "TAGENT"        # alphanumeric service code

#
#————————————————————————————————#
*MACHINES                                   #
#————————————————————————————————#
#
sheridan    LMID=SITE1
            ROOTDIR="/usr/people/informix/BENCH/TUXEDO"

            APPDIR="/usr/people/informix/BENCH/TPCC/db_client"
            TUXCONFIG="/usr/people/informix/BENCH/TPCC/tux/tuxconfig"
            ULOGPFX="/usr/people/informix/BENCH/TPCC/log/ULOG"
#
#————————————————————————————————#
*GROUPS
#
#————————————————————————————————#
#
group1          GRPNO=1                 # server group number
                LMID=SITE1              # logical machine identifier
```

```
group2          GRPNO=2                 # server group
number
                LMID=SITE1      # logical machine identi-
fier
group3          GRPNO=3                 # server group
number
                LMID=SITE1      # logical machine identi-
fier
group4          GRPNO=4                 # server group
number
                LMID=SITE1      # logical machine identi-
fier
group5          GRPNO=5                 # server group
number
                LMID=SITE1      # logical machine identi-
fier

#               OPENINFO=NONE# resource mgr. info
#               CLOSEINFO=""    # resource mgr. info
#
#————————————————————————————————#
#*NETWORK                                                #
#————————————————————————————————#
#
#SITE1   NADDR="0x000211119E3A0C84"
#       NLSADDR="0x000277779E3A0C84"
#       BRIDGE="/dev/tcp"
#————————————————————————————————#
*SERVERS                                                 #
#————————————————————————————————#
#
#DEFAULT:       SRVGRP=GROUP1           # server group
DEFAULT:        REPLYQ=Y                # reply queue,
Y=yes, N=no
#DEFAULT:       MAXGEN=5                # re-startable
times
#DEFAULT:       RESTART=Y               # re-start sta-
tus, Y=yes, N=no
#DEFAULT:       CLOPT="-A"              # booted op-
tions
#

# new_order servers
tpccfs  SRVGRP=group1
        CLOPT="-s NEWO_SVC — -n101 -csheridan -
Snewo_server -hvldb-1"
        RQADDR=neworder SRVID=101 MIN=45 MAX=50

# payment servers
tpccfs  SRVGRP=group2
        CLOPT="-s PMT_SVC — -n201 -csheridan -
Spmt_server -hvldb-1"
        RQADDR=payment SRVID=201 MIN=15 MAX=20

# order_status servers
tpccfs  SRVGRP=group3
        CLOPT="-s ORDS_SVC — -n301 -csheridan -
Sords_server -hvldb-1"
        RQADDR=orderstat SRVID=301 MIN=8 MAX=13

# delivery servers
tpccfs  SRVGRP=group4
        CLOPT="-s DVRY_SVC — -n401 -csheridan -
Sdvry_server -hvldb-1"
#       RQADDR=delivery SRVID=401 MIN=14 MAX=19

# stock_level servers
tpccfs  SRVGRP=group5
        CLOPT="-s STKL_SVC — -n501 -csheridan -
Sstck_server -hvldb-1"
        RQADDR=stocklevel SRVID=501 MIN=20 MAX=25

#
#
#————————————————————————————————#
*SERVICES                                                #
#————————————————————————————————#
#
```

INFORMIX-OnLine Configuration

```
#**************************************************************************
#
#                               INFORMIX SOFTWARE, INC.
#
#  Title:   onconfig.tpcc
#  Description: INFORMIX-OnLine Configuration Parameters
#
#**************************************************************************

# Root Dbspace Configuration

ROOTNAME        rootdbs         # Root dbspace name
ROOTPATH        /ifmx/Chk_RootDBS
                    # Path for device containing root dbspace
ROOTOFFSET      0               # Offset of root dbspace into device
(Kbytes)
ROOTSIZE        2090000         # Size of root dbspace (Kbytes)

# Disk Mirroring Configuration Parameters

MIRROR          1               # Mirroring flag (Yes = 1, No = 0)
MIRRORPATH                      # Path for device containing mirrored
root
MIRROROFFSET    0               # Offset into mirrored device (Kbytes)

# Physical Log Configuration

PHYSDBS         plog_ldbs01     # Location (dbspace) of physical
log
PHYSFILE        2080000         # Physical log file size (Kbytes)
                                    # Location (dbspace) of physi-
cal log

# Logical Log Configuration

LOGFILES        6               # Number of logical log files
LOGSIZE         2080000         # Logical log size (Kbytes)

# Diagnostics

MSGPATH         /usr/people/informix/BENCH/TPCC/online.log
```

```
                                        # System message log file path
CONSOLE         /usr/people/informix/BENCH/TPCC/console.log
                                        # System console message path
ALARMPROGRAM  /usr/people/informix/informix2/etc/log_full.sh
                        # Alarm program path

# System Archive Tape Device

TAPEDEV         /dev/null       # Tape device path
TAPEBLK         16              # Tape block size (Kbytes)
TAPESIZE        10240           # Maximum amount of data to put on tape (Kbytes)

# Log Archive Tape Device

LTAPEDEV        /dev/null       # Log tape device path
LTAPEBLK        16              # Log tape block size (Kbytes)
LTAPESIZE       10240           # Max amount of data to put on log tape (Kbytes)

# Optical

STAGEBLOB                       # INFORMIX-OnLine/Optical staging area

# System Configuration

SERVERNUM       0               # Unique id corresponding to a OnLine instance
DBSERVERNAME    tpcc            # Name of default database server
DBSERVERALIASES                 # List of alternate dbservernames
NETTYPE         ipcshm,6,31,CPU # Override sqlhosts nettype parameters
#NETTYPE        ipcshm,23,31,CPU # Override sqlhosts nettype parameters
DEADLOCK_TIMEOUT 60             # Max time to wait of lock in distributed env.
RESIDENT        0               # Forced residency flag (Yes = 1, No = 0)

MULTIPROCESSOR  1               # 0 for single-processor, 1 for multi-processor
NUMCPUVPS       6               # Number of user (cpu) vps
#NUMCPUVPS      23              # Number of user (cpu) vps
SINGLE_CPU_VP   0               # If non-zero, limit number of cpu vps to one

NOAGE           1               # Process aging
AFF_SPROC       1               # Affinity start processor
#AFF_NPROCS     23              # Affinity number of processors
AFF_NPROCS      6               # Affinity number of processors

# Shared Memory 0arameters

LOCKS           90000           # Maximum number of locks
BUFFERS         4600000         # Maximum number of shared buffers
#BUFFERS        3000000         # Maximum number of shared buffers
NUMAIOVPS       1               # Number of IO vps
PHYSBUFF        256             # Physical log buffer size (Kbytes)
LOGBUFF         256             # Logical log buffer size (Kbytes)
LOGSMAX         10              # Maximum number of logical log files
CLEANERS        366             # Number of buffer cleaner processes
#CLEANERS       128             # Number of buffer cleaner processes
SHMBASE         0x0             # Shared memory base address
SHMVIRTSIZE     520000          # initial virtual shared memory segment size
SHMADD          32000           # Size of new shared memory segments (Kbytes)
SHMTOTAL        0               # Total shared memory (Kbytes). 0=>unlimited
CKPTINTVL       1800            # Check point interval (in sec)
#LRUBUFFS                       4
LRUS            68              # Number of LRU queues
LRU_MIN_DIRTY   4               # LRU percent dirty end cleaning limit
LRU_MAX_DIRTY   5               # LRU percent dirty begin cleaning limit
#LRU_MIN_DIRTY  11              # LRU percent dirty end cleaning limit
#LRU_MAX_DIRTY  12              # LRU percent dirty begin cleaning limit
LTXHWM          50              # Long transaction high water mark percentage
LTXEHWM         60              # Long transaction high water mark (exclusive)
TXTIMEOUT       0x12c           # Transaction timeout (in sec)
STACKSIZE       32              # Stack size (Kbytes)

# System Page Size
# BUFFSIZE - OnLine no longer supports this configuration parameter.
#       To determine the page size used by OnLine on your platform
#       see the last line of output from the command, 'onstat -b'.


# Recovery Variables
# OFF_RECVRY_THREADS:
# Number of parallel worker threads during fast recovery or an offline restore.
# ON_RECVRY_THREADS:
# Number of parallel worker threads during an online restore.

OFF_RECVRY_THREADS 10           # Default number of offline worker threads
ON_RECVRY_THREADS 10            # Default number of online worker threads

# Data Replication Variables
# DRAUTO: 0 manual, 1 retain type, 2 reverse type
DRAUTO          0               # DR automatic switchover
DRINTERVAL      30              # DR max time between DR buffer flushes (in sec)
DRTIMEOUT       30              # DR network timeout (in sec)
DRLOSTFOUND     /usr/people/informix/etc/dr.lostfound # DR lost+found file path

# Read Ahead Variables
RA_PAGES        0               # Number of pages to attempt to read ahead
RA_THRESHOLD    0               # Number of pages left before next group

# DBSPACETEMP:
# OnLine equivalent of DBTEMP for SE. This is the list of
```

```
dbspaces
# that the OnLine SQL Engine will use to create temp tables etc.
# If specified it must be a colon separated list of dbspaces that
exist
# when the OnLine system is brought online.  If not specified, or
if
# all dbspaces specified are invalid, various ad hoc queries will
create
# temporary files in /tmp instead.

DBSPACETEMP     rootdbs         # Default temp dbspaces

# DUMP*:
# The following parameters control the type of diagnostics infor-
mation which
# is preserved when an unanticipated error condition (assertion
failure) occurs
# during OnLine operations.
# For DUMPSHMEM, DUMPGCORE and DUMPCORE 1 means
Yes, 0 means No.

DUMPDIR         /tmp            # Preserve diagnostics in this directory
DUMPSHMEM       0               # Dump a copy of shared memory
DUMPGCORE       0               # Dump a core image using 'gcore'
DUMPCORE        0               # Dump a core image (Warning:this
aborts OnLine)
DUMPCNT         0               # Number of shared memory or gcore
dumps for
                                # a single user's session

# ADT*
# The following parameters control the type and level of secure
auditing
# present in the OnLine system. By default, ADTMODE is 0 and
auditing
# is disabled


FILLFACTOR      90              # Fill factor for building indexes

# method for OnLine to use when determining current time
USEOSTIME       0               # 0: use internal time(fast), 1: get time
from OS(slow)

# Parallel Database Queries (pdq)
                #                       OFF => 0, LOW => 1,
HIGH => 100
MAX_PDQPRIORITY 100             # Maximum allowed pdqpriority
DS_MAX_QUERIES  2               # Maximum number of decision
support queries
DS_TOTAL_MEMORY                 # Decision support memory
(Kbytes)
DS_MAX_SCANS    3               # Maximum number of decision
support scans
#BTAPPENDERS     7
DATASKIP        off             # List of dbspaces to skip

# OPTCOMPIND
# 0 => Nested loop joins will be preferred (where
#       possible) over sortmerge joins and hash joins.
# 1 => If the transaction isolation mode is not
#       "repeatable read", optimizer behaves as in (2)
#       below.  Otherwise it behaves as in (0) above.
# 2 => Use costs regardless of the transaction isolation
#       mode.  Nested loop joins are not necessarily
#       preferred.  Optimizer bases its decision purely
#       on costs.
OPTCOMPIND      0               # To hint the optimizer
ONDBSPACEDOWN   0               # Dbspace down option: 0 =
CONTINUE, 1 = ABORT, 2 = WAIT
LBU_PRESERVE    0               # Preserve last log for log backup
OPCACHEMAX      128             # Maximum optical cache size
(Kbytes)
BAR_ACT_LOG     /tmp/bar_act.log
BAR_MAX_BACKUP  0
BAR_RETRY       1
BAR_NB_XPORT_COUNT 10
BAR_XFER_BUF_SIZE 31
HETERO_COMMIT   0
```

Appendix D - Disk Sto

The calculations used to determine the storage requiremen
logical log and the 180-day space are contained in this a
capacity of each of the 4.2 is 4,509,863,424 bytes (4.200

Table	Rows	Data	Index	Bitmap	Free	Daily growth	5% space
Warehouse	2,200	2,200	2	1	49		110
District	22,000	2,201	2	2	49		110
Item	100,000	4,762	1	3	36		238
History	66,000,000	1,736,843	0	432	347,575	319,775	0
Order	66,000,000	1,047,620	774,448	460	1,085,975	192,917	0
Customer	66,000,000	22,000,000	1,646,376	413	58,266		1,182,339
New Order	19,800,000	117,859	211,839	84	63,968		16,489
Order Line	659,918,692	30,000,004	1	7,473	6,665,418	5,523,384	0
Stock	220,000,000	36,666,667	1	9,108	5,826		1,833,789
Totals		**91,578,156**	**2,632,670**	**17,976**	**8,227,162**	**6,036,075**	**3,033,076**

New Order tpmC: 25,309.20

New Order tpmC per Warehouse: 11.50

Informix Disk Space Specifications			
DBspaces	**Chunks**	**Chunk Size**	**Allocated**
rootdbs	1	1,045,000	1,045,000
customer	71	varied	25,694,500
history	3	700,000	2,100,000
new order	6	varied	585,000
misc. tables	2	varied	10,000
order	21	varied	3,076,500
orderline	70	524,000	36,680,000
stock	192	191,500	36,768,000
TOTAL	**366**		**105,959,000**

Misc. tables = Warehouse, District, and Item

Dynamic space	32,792,832	Sum of Data + Bitmap for Order, Order_Line and History
Static space	64,469,046	Sum of all data,index,bitmap + 5% - above dynamic space
Free space	8,697,122	Total space allocated to Informix - dynamic and static
Daily growth	6,036,075	(Dynamic space/W * 62.5)* tpmC
Daily spread	(356,991)	Free space - 1.5 * Daily growth (zero if negative)
180-day spread	0	Informix may be configured so that the daily spread is zero
180 day space	1,150,962,587	Static space + 180 (daily growth + daily spread)
180 day in GB	**2,195.29**	

Eight-Hour Recovery Log Calculation				
		Pages per	Eight-Hour Utilization	
Pages Written	**Transactions**	**Transaction**	**Pages**	**GB**
392,012	252,115	1.555	18,889,494.290	36.029

Appendix E - RTE Scri

Transaction Selection

```
/* EMPOWER V3.1.3 */
/* TERM=vt100 */
/* capture test telnet:hp_t500  */
/* compose test telnet:hp_t500  */

#include "/source/empower/h/vt100.h"
#include "../incl/rte_tpcc.h"

int     warehouse_id, user_id, stkl_d_id, trans_type, option;
char    buf[50];
extern int set_state();
extern int tpcc_state;


   Timeout(300, EXIT);          /* What to do if Rcv() takes too long */
   Unset(NOTIFY);               /* Don't display warnings. Mon will show them */
   Term(ZOOM, VT100|LINES24|AUTOWRAP);
   Set(BDELAY);                 /* Use block delay for keying time */
   tpcc_state= STATE_BEGIN;
   Signal(set_state);

   check_parms(argc,argv);      /* check for parameters */

   Rcv(": ");                   /* wait for login */
   login(argv[4], argv[5]);     /*  and login */

   user_id= atoi(argv[3]); /* get the warehouse id */
   Seed(user_id);               /* Seed random number generator */
   if (user_id == 0)       /* only user zero process must initialize */
      initbench();

   warehouse_id= (user_id-1)/DISTRICTS_PER_WAREHOUSE + 1;
   stkl_d_id = (user_id-1)%DISTRICTS_PER_WAREHOUSE + 1;

   if (warehouse_id <0){
      sprintf(buf,"Illegal Warehouse = %+d",warehouse_id);
      Log(&buf);
      Xmit("exit^M");   /* Exit STATE_STOP has been received */
      Wait(2);

      exit(0);
```

```
  }
   sprintf(buf,"W%dD%d",warehouse_id, stkl_d_id);
  Note(&buf);                /* give the Warehouse ID to the
monitor */

  sprintf(buf,"exec ./client %d", user_id);
  Mxmit(&buf, "^M", "");
  Rcv("||");                 /*  and wait for initialization */
  Suspend();                 /* suspend until a resume signal
  Note("STARTING");
                             activated from mon or mix */
  set_state();               /* set the current banchmark state.
                             The note column in mon, will show
                             the state of the benchmark. */


  Beginscenario("Tpc-c Benchmark");
  while ( tpcc_state !=  STATE_STOP) {  /* do until state is stop
*/

     trans_type= Range(1,100000);
     if (trans_type < 4060) option=STOCKLEV;          /* stocklev
4.06% */
     else if (trans_type < 8120) option=DELIVERY;     /* delivery
4.06% */
     else if (trans_type < 12180) option=ORDSTAT;     /* ordstat
4.06% */
     else if (trans_type < 55380) option=PAYMENT;     /*
payment 43.2% */
     else option=NEWORD;                    /* neword 43+% */


     switch (option) {
        case NEWORD:
          Thinktne(0,19.1,120);
          rte_neword(warehouse_id);
          break;
        case PAYMENT:
          Thinktne(0,19.1,120);
          rte_payment(warehouse_id);
          break;
        case ORDSTAT:
          Thinktne(0,9.8,100);
          rte_ordstat(warehouse_id);
          break;
        case DELIVERY:
          Thinktne(0,4.8,50);
          rte_delivery(warehouse_id);
          break;
        case STOCKLEV:
          Thinktne(0,4.8,50);
          rte_stocklev(warehouse_id, stkl_d_id);
          break;
        default:
          printf("*** ERROR ***, no such option exists. \n");
          break;
      }
  }
  Nametransaction("Stop");
  Xmit("6^M");          /* Stop the transactions */
  Wait(2);
Endscenario("Tpc-c Benchmark");
```

Input Data Generation

```
/* EMPOWER V3.1.3 */

#include "../incl/rte_tpcc.h"

int     first=0;
char    buf[1100], OL_S[200], logO[1100];
extern char *last_name_parts[10];

/*********************************************************
*       Generate a new order transaction                 *
*********************************************************/
rte_neword(warehouse_id)
int warehouse_id;
{
struct rte_neworder  neworder;
int     offset, i, length;

Beginsource("rte_trans");
Beginfunction("New Order transactions");

  /* Get New Order Screen */
    Set(AUTOTHINK);              /* Turn think time on.  */
    Nametransaction("NewOrdMenue");   /* <== transaction type
*/
    sprintf(&buf,"%01d", NEWORD );
    Mxmit(&buf, "^M", "");          /* <== send the buffer  */
                              /* wait for reply    */
    Mrcv( "||^[=!<^[=!<", "");

    Unset(AUTOTHINK);            /* turn thinktime off. */
    Sleep(18);                   /* Keying time */

 /* Generate transaction */
   neworder.s_W_ID = warehouse_id;
   neworder.s_D_ID =
Range(1,DISTRICTS_PER_WAREHOUSE);
   neworder.s_C_ID = NUrand_val(A_C_ID, 1,
CUSTOMERS_PER_DISTRICT, C_C_ID);
   neworder.s_O_OL_CNT = Range(5,15);
   neworder.s_temp_cnt = neworder.s_O_OL_CNT;
   sprintf(OL_S," OrderLines %d",neworder.s_O_OL_CNT);
   Log(&OL_S);
   neworder.s_all_local = 1;     /* Let us say all orders are local
*/
#ifdef DEBUG
   sprintf(logO," NEWORD s_D_ID= %02d s_C_ID= %04d
neworder.s_O_OL_CNT= %02d \
   neworder.s_all_local= %01d", neworder.s_D_ID,
neworder.s_C_ID, \
   neworder.s_O_OL_CNT, neworder.s_all_local);
   Log(&logO);
#endif

   /* generate all order lines */
   for (i=0; i < neworder.s_O_OL_CNT; i++) {
      if ( Range(1,10000) > 105 )
          neworder.items[i].s_OL_SUPPLY_W_ID =
neworder.s_W_ID;
      else {
         neworder.s_all_local = 0;  /* Not all orders are local */
         while ((neworder.items[i].s_OL_SUPPLY_W_ID =
Range(1,WAREHOUSES)) ==
```

```
          neworder.s_W_ID);
      }
      neworder.items[i].s_OL_I_ID= NUrand_val(A_OL_I_ID, 1,
ITEMS, C_OL_I_ID);
      neworder.items[i].s_OL_QUANTITY = Range(1,10);
#ifdef DEBUG
      sprintf(logO,"s_OL_SUPPLY_W_ID= %04d s_OL_I_ID=
%06d s_OL_QUANTITY= %02d"
,
      neworder.items[i].s_OL_SUPPLY_W_ID,
neworder.items[i].s_OL_I_ID,
      neworder.items[i].s_OL_QUANTITY);
      Log(&logO);
#endif
   }
   /* is it a rollback transaction ? */
   i = Range(1,10000);
   if (i < 102) {             /* Yes! */
       neworder.items[neworder.s_O_OL_CNT -1].s_OL_I_ID =
999999;
       if (neworder.s_all_local)
          Nametransaction("NewOrderRollback");   /* <==
transaction type */
       else
          Nametransaction("NewOrderRollRem");   /* <==
transaction type */
    }
    else {
    if (neworder.s_all_local)
        Nametransaction("NewOrder");   /* <== transaction type
*/
    else
        Nametransaction("NewOrderRemote");   /* <== transac-
tion type */
    }
   /* prepare the empower buffer */
   offset= length =0;

    sprintf(&buf[offset],"%02d^I%04d^I",
      neworder.s_D_ID, neworder.s_C_ID);

   offset= 10;   /* Next offset into empower buffer. */
   length= 18;   /* Length of the next bugger input. */

   for (i=0; i < neworder.s_O_OL_CNT; i++){
      sprintf(&buf[offset],"%04d^I%06d^I%02d%^I",
neworder.items[i].s_OL_SUPPLY
_W_ID,
         neworder.items[i].s_OL_I_ID,
neworder.items[i].s_OL_QUANTITY );
      offset= offset + length;  /* Next offset into empower buffer.
*/
    }

   /***** Do the neworder transaction *****/
   Mxmit(&buf, "^M", "");       /* <== send the buffer  */
   Mrcv( "data.^[=6 ^[=7", "...^[=6 ^[=7i", "||", "");
   Endfunction("New Order transactions");
}

/*----------------------------------------------------------*/
/*      Generate a payment transaction                      */
/*----------------------------------------------------------*/
```

```
rte_payment( warehouse)
int warehouse;
{
static int      i, j, type;
struct rte_payment payment;

Beginfunction("Payment transactions");


if (first == 0 ) {
       init_name_part();
       first++;
#ifdef ACID
   for( j=0; j < 3000; j++) {
      i = NUrand_val(A_C_LAST,1, 10, C_C_LAST) - 1;
      strcpy(payment.s_C_LAST, last_name_parts[i]);
      i = NUrand_val(A_C_LAST,1, 10, C_C_LAST) - 1;
      strcat(payment.s_C_LAST, last_name_parts[i]);
      i = NUrand_val(A_C_LAST,1, 10, C_C_LAST) - 1;
      strcat(payment.s_C_LAST, last_name_parts[i]);
      sprintf(buf,"ACID C_LAST= %s\n", payment.s_C_LAST);
      Log(&buf);
   }
#endif
}
  /* Get Payment Screen */
    Set(AUTOTHINK);               /* Turn think time on.  */
    Nametransaction("PayMenue");   /* <== transaction type */
    sprintf(&buf,"%01d", PAYMENT );
    Mxmit(&buf, "^M", "");          /* <== send the buffer  */
    Mrcv( "...^[=6 ^[=7i", "||^[=#S^[=#S", "");

  /* Generate Payment Transaction */
   payment.s_W_ID = warehouse;
   payment.s_D_ID =
Range(1,DISTRICTS_PER_WAREHOUSE);
   payment.s_H_AMOUNT = Range(1,500000);  /* 5000.00 */
   if ( Range(1,1000) <= 850 ) {
      type= 0;
      payment.s_C_W_ID = payment.s_W_ID;
      payment.s_C_D_ID = payment.s_D_ID;
    }
    else {
      type= 1;
      payment.s_C_D_ID = Range(1,
DISTRICTS_PER_WAREHOUSE);
       while ((payment.s_C_W_ID = Range(1,WAREHOUSES))
==  payment.s_W_ID);
     }

   if (Range(1,1000) < 400) {
      payment.s_C_ID =
NUrand_val(A_C_ID,1,CUSTOMERS_PER_DISTRICT,
C_C_ID);
      payment.s_C_LAST[0] = (char)NULL;
      sprintf(buf,"%02d^I%04d^I%04d^I%02d^I^I%06d",
payment.s_D_ID,
       payment.s_C_ID, payment.s_C_W_ID,
payment.s_C_D_ID, payment.s_H_AMOUNT);
#ifdef DEBUG
    sprintf(logO," PAYMENT s_D_ID= %d s_C_ID= %d
s_C_W_ID= %04d s_C_D_ID= %02d \
       s_H_AMOUNT= %d", payment.s_D_ID, payment.s_C_ID,
```

```
payment.s_C_W_ID,
       payment.s_C_D_ID, payment.s_H_AMOUNT );
    Log(&logO);
#endif
   }
   else
   {
      if ( type == 0 )
          type= 2;
      else type= 3;
      payment.s_C_ID = 0;
      i = NUrand_val(A_C_LAST,1, 10, C_C_LAST) - 1;
       strcpy(payment.s_C_LAST, last_name_parts[i]);
      i = NUrand_val(A_C_LAST,1, 10, C_C_LAST) - 1;
       strcat(payment.s_C_LAST, last_name_parts[i]);
      i = NUrand_val(A_C_LAST,1, 10, C_C_LAST) - 1;
       strcat(payment.s_C_LAST, last_name_parts[i]);
      payment.s_all_local = 1;
      sprintf(buf,"%02d^I^I%04d^I%02d^I%15s^I%06d",
        payment.s_D_ID, payment.s_C_W_ID,
        payment.s_C_D_ID, payment.s_C_LAST,
payment.s_H_AMOUNT);
#ifdef DEBUG
    sprintf(logO," PAYMENT  s_D_ID= %d s_C_ID= %d \
       s_C_W_ID= %04d s_C_D_ID= %02d s_H_AMOUNT= %d
s_C_LAST= %s",
        payment.s_D_ID, payment.s_C_ID, payment.s_C_W_ID,
        payment.s_C_D_ID, payment.s_H_AMOUNT ,
payment.s_C_LAST);
    Log(&logO);
#endif
    }
    if ( type == 0 ) Nametransaction("PaymentHoCid");
    if ( type == 1 ) Nametransaction("PaymentReCid");
    if ( type == 2 ) Nametransaction("PaymentHoLast");
    if ( type == 3 ) Nametransaction("PaymentReLast");

    /***** Do the Payment transaction *****/
    Unset(AUTOTHINK);         /* Turn think time off  */
    Sleep(3);                 /* Keying time */
    Mxmit(&buf, "^M", ""); /*  send message to the SUT */
    Mrcv(  "...^[=6 ^[=7i||", "||", "");
     Endfunction("Payment transactions");
}

/*-------------------------------------------------------*/
/*      Generate order status transaction                */
/*-------------------------------------------------------*/
rte_ordstat(warehouse)
int warehouse;
{
static int i, j;
struct rte_ordsta ordsta;

Beginfunction("Order Status transactions");

if (first == 0 ) {
        init_name_part();
        first++;
#ifdef ACID
   for( j=0; j < 3000; j++) {
      i = NUrand_val(A_C_LAST,1, 10, C_C_LAST) - 1;
       strcpy(ordsta.s_C_LAST, last_name_parts[i]);
      i = NUrand_val(A_C_LAST,1, 10, C_C_LAST) - 1;
       strcat(ordsta.s_C_LAST, last_name_parts[i]);
      i = NUrand_val(A_C_LAST,1, 10, C_C_LAST) - 1;
       strcat(ordsta.s_C_LAST, last_name_parts[i]);
      sprintf(buf,"ACID C_LAST= %s\n", ordsta.s_C_LAST);
      Log(&buf);
   }
#endif
}

  /* Get Order Status Screen */
    Set(AUTOTHINK);               /* Turn think time on.  */
    Nametransaction("OrdStatMenue");   /* <== transaction type
*/
    sprintf(&buf,"%01d", ORDSTAT );
    Mxmit(&buf, "^M", "");          /* <== send the buffer  */
    Mrcv( "||^[=!<^[=!<",  "");

  /* Generate IOrder Status transaction */
   ordsta.s_W_ID = warehouse;
   ordsta.s_D_ID =
Range(1,DISTRICTS_PER_WAREHOUSE);
   if (Range(1,1000) < 400) {
        Nametransaction("OrderStatus ");      /* transaction name
*/
        ordsta.s_C_ID =
NUrand_val(A_C_ID,1,CUSTOMERS_PER_DISTRICT,
C_C_ID);
        ordsta.s_C_LAST[0] = (char)NULL;
   }
   else
   {
        Nametransaction("OrderStatusLast");      /* transaction
name */
        ordsta.s_C_ID = 0;
        i = NUrand_val(A_C_LAST,1, 10, C_C_LAST) - 1;
         strcpy(ordsta.s_C_LAST, last_name_parts[i]);
        i = NUrand_val(A_C_LAST,1, 10, C_C_LAST) - 1;
         strcat(ordsta.s_C_LAST, last_name_parts[i]);
        i = NUrand_val(A_C_LAST,1, 10, C_C_LAST) - 1;
         strcat(ordsta.s_C_LAST, last_name_parts[i]);
   }
#ifdef DEBUG
    sprintf(logO," ORDSTAT= s_D_ID= %d s_C_ID= %04d
s_C_LAST= %s",
      ordsta.s_D_ID, ordsta.s_C_ID, ordsta.s_C_LAST);
    Log(&logO);
#endif
    if (ordsta.s_C_ID > 0 )
      sprintf(buf,"%02d^I^I%04d", ordsta.s_D_ID, ordsta.s_C_ID);
    else
      sprintf(buf,"%02d^I^I%15s",
      ordsta.s_D_ID, ordsta.s_C_LAST);

      /***** Do the Order Status transaction *****/
       Unset(AUTOTHINK);                  /* Turn think time off
*/
       Sleep(2);                          /* Keying time */
       Mxmit(&buf, "^M", "");             /* send to the SUT */
       Mrcv(  "...^[=6 ^[=7i", "||", "");
       Endfunction("Order Status transactions");
}
```

```
/*————————————————————————————————————*/
/*      Generate stock level transaction                */
/*————————————————————————————————————*/
rte_stocklev(warehouse, district)
int warehouse, district;
{
struct rte_stocklev stocklev;
Beginfunction("Stock Level transactions");

  /* Get Stock Level Screen */
    Set(AUTOTHINK);              /* Turn think time on.  */
    Nametransaction("StocklevMenue");   /* <== transaction type
*/
    sprintf(&buf,"%01d", STOCKLEV );
    Mxmit(&buf,  "");          /* <== send the buffer  */
    Mrcv( "||^[=#7",  "");

 /* Generate the Stock Level transaction */
   /* The SUT already has this informiation
   stocklev.s_W_ID = warehouse;
   stocklev.s_D_ID = district;
   */
   stocklev.s_threshold = Range(10,20);

#ifdef DEBUG
   sprintf(logO," STOCKLEV s_threshold= %d",
stocklev.s_threshold);
   Log(&logO);
#endif
   sprintf(buf,"%02d", stocklev.s_threshold);

   /***** Do the Stock Level transaction *****/
   Unset(AUTOTHINK);                    /* Turn think time off.  */
   Sleep(2);          /* Keying time */
   Nametransaction("StockLevel ");      /* transaction name */
   Mxmit(&buf, "^M", "");               /* send to SUT */
   Mrcv(  "...^[=6 ^[=7i", "||", "");
   Endfunction("Stock Level transactions");
}

/*————————————————————————————————————*/
/*             Generate delivery transaction           */
/*————————————————————————————————————*/
rte_delivery(warehouse)
int warehouse;
{
struct rte_delivery delivery;
Beginfunction("Delivery transactions");

  /* Get Delivery Screen */
    Set(AUTOTHINK);              /* Turn think time on.  */
    Nametransaction("DeliveMenue");   /* <== transaction type */
    sprintf(&buf,"%01d", DELIVERY );
    Mxmit(&buf,  "");          /* <== send the buffer  */
    Mrcv( "||^[=#0", "");

   /* The SUT already has this information
   delivery.s_W_ID = warehouse;
   */
   delivery.s_O_CARRIER_ID = Range(1,10);

#ifdef DEBUG
   sprintf(logO," DELIVERY s_O_CARRIER_ID= %d",
delivery.s_O_CARRIER_ID);
   Log(&logO);
#endif
   sprintf(buf,"%02d", delivery.s_O_CARRIER_ID);

   /***** Do the Delivery transaction *****/
   Unset(AUTOTHINK);                    /* Turn think time on.  */
   Sleep(2);                          /* Keying time */
   Nametransaction("Delivery");         /* transaction name */
   Mxmit(&buf, "^M", "");               /* send to SUT */
   Mrcv(  "...^[=6 ^[=7i", "||",  "");
   Endfunction("Delivery transactions");
Endsource();
}
```

Appendix F - Third-Party Quot

Computer Modules, Inc.
2400 WALSH AVENUE
SANTA CLARA, CA 95051
Phone: 408.496.1881
Fax: 408.496.1886
e-mail: info@compmod.com
Internet: www.compmod.com

Computer Modules, Inc
2350 Walsh Avenue
Santa Clara, CA 95051
(408) 496-1881 - FAX (408) 496-1886

Silicon Graphics
Attn: Mr. Cuong Tran
(415) 962-8404

Quote #	981 BR
Date:	4-2-97
Inquiry Date:	3-15-97
Prop.Ship Date	T.B.D
Terms:	Net 30
F.O.B.	Santa Clara
Sales Person	Les Zoltan
Shipped Via	UPS

CMI is pleased to quote on your request for:

		Price Per.
900-1300	Acculan 24 port Hub-Ultra Slim Line P/N: AL-HUB 24L	$212.50

Specifications: Per Acculan
Ultra Slim Line Specs
Enclosed.

Warranty: 5 Year return to Factory

Quote Valid for 60 days

By:

BEA ENTERPRISE MIDDLEWARE SOLUTIONS

August 20, 1996

Mr. Cuong Tran
Member of Technical Staff
System Software Development
Silicon Graphics Computer Systems
2011 N. Shoreline Blvd.
PO Box 7311
Mountain View, CA 94039-7311

Subject: Pricing for Tuxedo 6.1

Dear Mr. Tran;

This letter is to confirm our policy on Tuxedo versions' general availability. At present we are in the process of removing Tuxedo 4.2.2 (from BEA, ITI, IMC or Novell) from general availabilty. It is replaced by Tuxedo 6.1 from BEA Systems, Inc.

For purposes of TPC activity we recommend the use of Tuxedo 6.1 available through our Core Functionality Services (CFS) program. This is a replacement for version 4.2.2 or earlier. This is a genrally available product and should be denoted as Tuxedo 6.1 CFS in all publications. It is priced as shown below.

BEA Tuxedo Core Functionaty Services Program 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, RISC Uniprocessors	Unlimited	$3,000.00	$450.00	$660.00
Tier 2 - PC Servers with 3 or 4 CPUs, Midrange and UNIX Systems	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 (8 - 32 CPUs) and Mainframe Systems	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

Sincerely,

Lew Brentano
Director, Product Marketing
cc: Dennis Lyftogt, BEA Sales

1DIRECT Sales Quotation

Proposal To: **Melanie Fulton**
Company/Org: **Silicon Graphics**
Address: **2011 N. Shoreline Drive**

City/St/Zip: **Mountain View, CA 94039**
Telephone: **415-933-1995**
FAX: **415-964-0811**

Quote #: **MF0428** V2.2.5
Date: **04/28/97**
Reference:

End User:
Phone:
Fax:

Item	Qty	Vendor	Part Number	Description	$ List Price	% Disc.	$ Unit Price	$ Extended Price
				Server Hardware				
1	1	SGI	R1-H64002-4	Origin2000 Rack, 2XR10K, 195MHz, 4MB Cache, 64MB Mem on 1 Node brd, 4.5GB sys disk,CDROM,IRIX for 2 CPUs	100,000.00	30.0%	70,000.00	70,000.00
2	1	SGI	HU1-RL6402-4	Origin2000 Exp Rack w/Insert Module, 2XR10000 w/4MB Cache, 64MB Mem on 1 Node board, IRIX for 2 CPUs	85,000.00	30.0%	59,500.00	59,500.00
3	2	SGI	HU1-ML6402-4	Origin2000 Insert Module, 2XR10000 w/4MB Cache, 64MB Mem on 1 Node board, IRIX for 2 CPUs	65,000.00	30.0%	45,500.00	91,000.00
4	10	SGI	HU1-H6402-4	2XR10000 w/4MB Cache, 64MB Mem Node board, IRIX for 2 CPUs for Origin2000 & Onyx2	57,000.00	30.0%	39,900.00	399,000.00
5	12	SGI	OPN-64U1GB-8	First 1GB Memory (8 banks-16Mbit) for Origin2000 and Onyx2 Systems	76,800.00	30.0%	53,760.00	645,120.00
6	6	SGI	H4-N128	Additional 128MB Memory (one bank-16Mbit) for Origin2000, Origin200 and Onyx2 Systems	10,240.00	30.0%	7,168.00	43,008.00
7	2	SGI	OPN-64U128-1	First 128MB Memory (one bank-16Mbit) for Origin2000, Origin200 (2-4 cpus only) and Onyx2 Systems	5,120.00	30.0%	3,584.00	7,168.00
8	12	SGI	CR-64U128-1	Standard memory program	(19,200.00)	30.0%	(13,440.00)	(161,280.00)
9	6	SGI	CR-64U1GB-8	Standard memory program	(2,560.00)	30.0%	(1,792.00)	(10,752.00)
10	2	SGI	CR-N128	Standard memory program	(1,280.00)	30.0%	(896.00)	(1,792.00)
11	18	SGI	XT-SCSI-4P	4-port Ultra SCSI XIO card for Origin2000 and Onyx2 systems	3,000.00	30.0%	2,100.00	37,800.00
12	1	SGI	XT-FE-4TX-6A	XIO 4-Port Fast-Ethernet (100Base-Tx) Adapter with 6 Asyn.(460 kbps) Serial Ports. Origin2000 and Onyx2	6,000.00	30.0%	4,200.00	4,200.00
13	1	SGI	HU-IC-XP-32	Upgrade Xpress Links for 32 CPU systems	10,000.00	30.0%	7,000.00	7,000.00
14	1	SGI	HU-IC-17TO32	Upgrade CrayLink cables for 17 to 32 CPU configurations	15,000.00	30.0%	10,500.00	10,500.00
15	1	SGI	DK-LR2-003	Rack system and rack expansion chassis destination kit for Origin2000 and Onyx2, US/Canada	-		-	-
16	1	SGI	P-S-2TB	6 full vault XLs	1,268,400.00	30.0%	887,880.00	887,880.00
17	1	SGI	FTO-4GB	4.5GB SE Ultra SCSI System Disk for Origin2000 and Onyx2 systems	-		-	-
18	1	Wyse	WY-55	Wyse55 terminal			589.00	589.00
				Client Hardware				
19	26	SGI	CH-S200-2G64-N	Challenge S, 1 CPU, 200MHz, 64MB memory, 2GB disk	15,900.00	30.0%	11,130.00	289,380.00
20	26	SGI	CR-CHS-SC4PC	Standard credit for Challenge S	(6,200.00)	30.0%	(4,340.00)	(112,840.00)
21	26	Wyse	WY-55	Wyse55 Terminal			589.00	15,314.00
22	26	SGI	HU-M128S	128MB Memory upgrade for Indy, Challenge S, Indigo2, POWER Indigo2, ChallengeM, POWER Challenge M	4,000.00	30.0%	2,800.00	72,800.00
23	26	SGI	HU-M64A	64MB Memory upgrade for Indigo2,POWER Indigo2, Indy,Challenge S, Challenge M, & POWER CHALLENGE M	2,000.00	30.0%	1,400.00	36,400.00

RGF

Item	Qty	Vendor	Part Number	Description	$ List Price	% Disc.	$ Unit Price	$ Extended Price
				Server Software				
24	1	sgi	SC4-IRIX64-6.4	IRIX 6.4 S2MP	-		-	-
				Five-Year Support				
25	1	SGI	FC-EW5	Full Care Extended Warranty - 5 Years (for SGI products in items 1 - 17)	350,420.00	30.0%	245,294.00	245,294.00
26	1	SGI	FC-EW5	Full Care Extended Warranty - 5 Years (for item 19)	6,425.00	30.0%	4,497.50	4,497.50

Delivery:	Delivery TBD; Ship via Drop Ship		
Notes:	Above configuration is generally available and price are guaranteed for 60 days from quotation date.	**TOTAL**	**$ 2,639,786.50**

Terms and Conditions: ; Quote Valid for 60 days

Terms and Conditions for this sales quotation, unless modified above, are: payment Net 30 (upon approved credit); taxes and freight not included; FOB Origin; 15% restocking fee (for approved returns of unused products); returns not accepted without RMA number; ship via UPS Ground.

This document is confidential and is not to be forwarded to third parties without the written consent of 1DIRECT. Quote valid for 30 days.

Bob Thompson
1DIRECT
700 E. El Camino Real, #250
Mountain View, CA 94040

Tel: 415-335-1765
Fax: 415-335-1790
Pager: 888-752-0958
bob@1direct.com

267