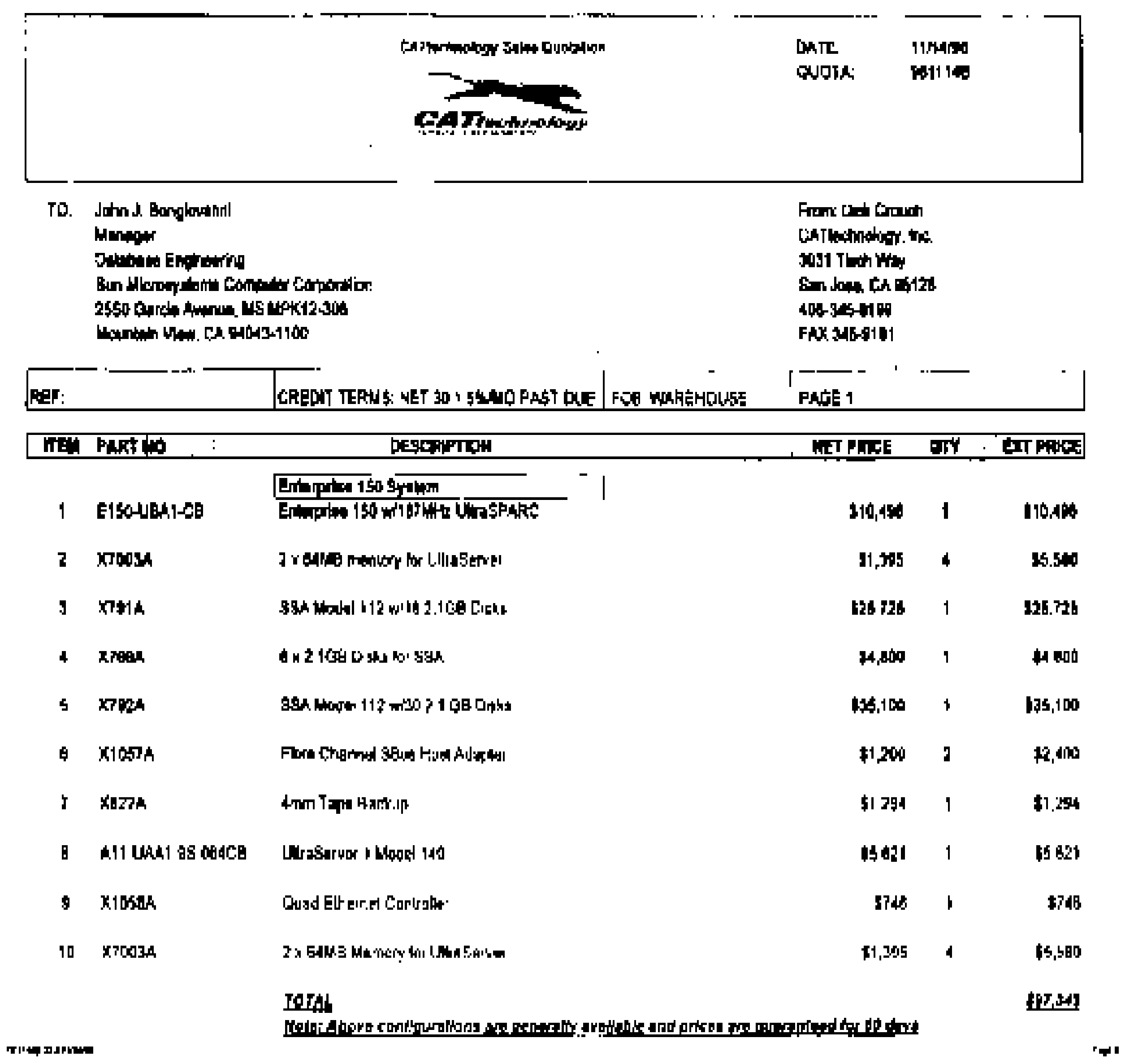


CATechnology Sales Quotation

DATE: 11/14/96
QUOTA: 9611140

TO: John J. Bonglovanni
Manager
Database Engineering
Sun Microsystems Computer Corporation
2550 Garcia Avenue, MS MPK12-306
Mountain View, CA 94043-1100

From: Dale Crouch
CATechnology, Inc.
3031 Tisch Way
San Jose, CA 95128
408-345-9190
FAX 345-9191

REF:	CREDIT TERMS: NET 30 1.5%/MO PAST DUE	FOB WAREHOUSE	PAGE 1

ITEM	PART NO	DESCRIPTION	NET PRICE	QTY	EXT PRICE
		Enterprise 150 System			
1	E150-UBA1-CB	Enterprise 150 w/167MHz UltraSPARC	$10,496	1	$10,496
2	X7003A	2 x 64MB memory for UltraServer	$1,395	4	$5,580
3	X791A	SSA Model 112 with 2.1GB Disks	$28,728	1	$28,728
4	X788A	6 x 2.1GB Disks for SSA	$4,800	1	$4,800
5	X792A	SSA Model 112 w/30 2.1 GB Disks	$35,100	1	$35,100
6	X1057A	Fibre Channel SBus Host Adapter	$1,200	2	$2,400
7	X827A	4mm Tape Backup	$1,294	1	$1,294
8	A11 UAA1 9S 064CB	UltraServer 1 Model 140	$5,621	1	$5,621
9	X1058A	Quad Ethernet Controller	$748	1	$748
10	X7003A	2 x 64MB Memory for UltraServer	$1,395	4	$5,580
		TOTAL			$97,343

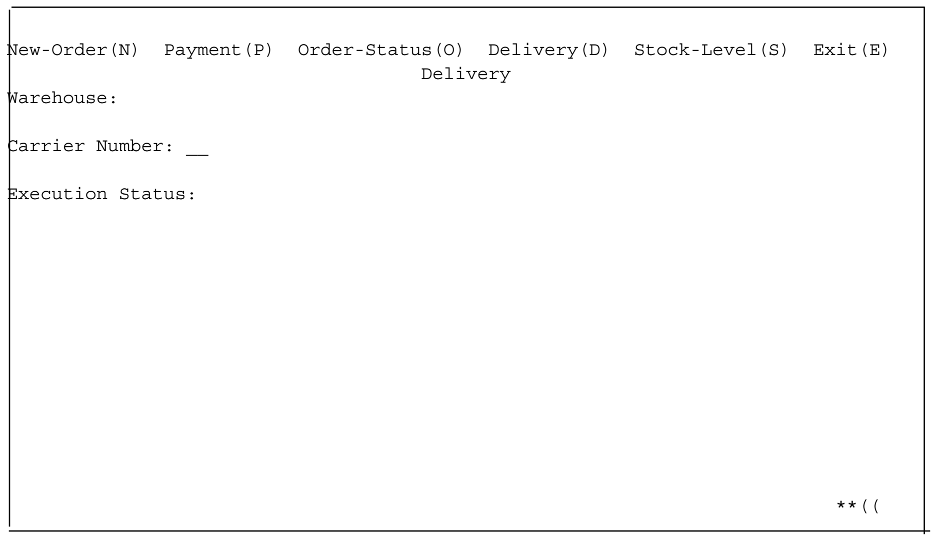
Note: Above configurations are generally available and prices are guaranteed for 60 days

11-14-1996 12:55PM FROM CAT TECHNOLOGY 408 345 9191 P.3

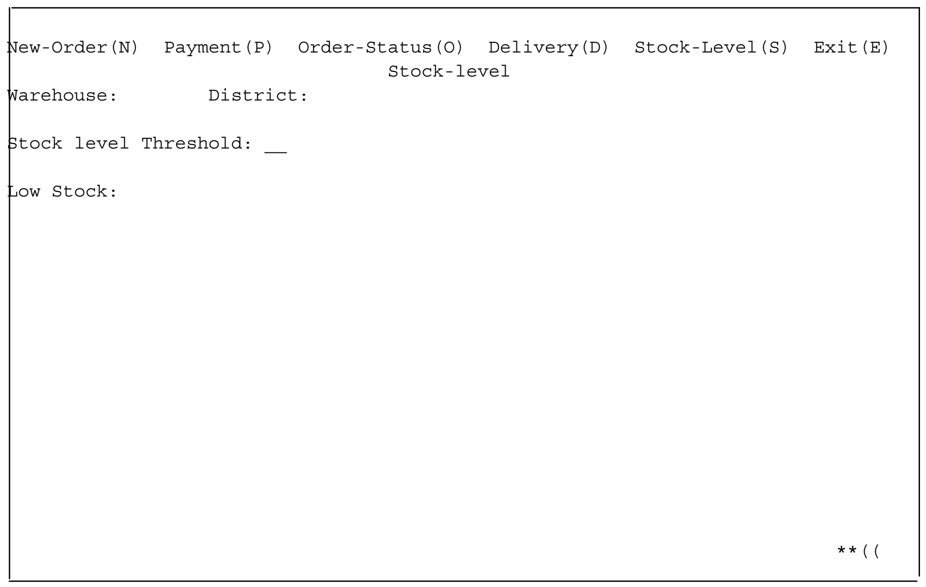
Appendix G: Price Quotes

G

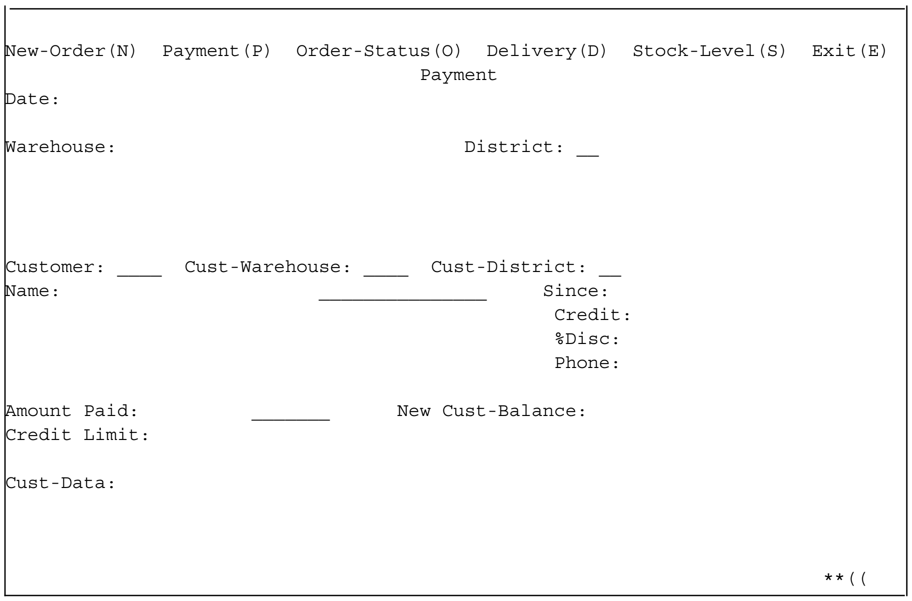
The following pages contain the pricing quotes for the hardware and software included in this FDR.



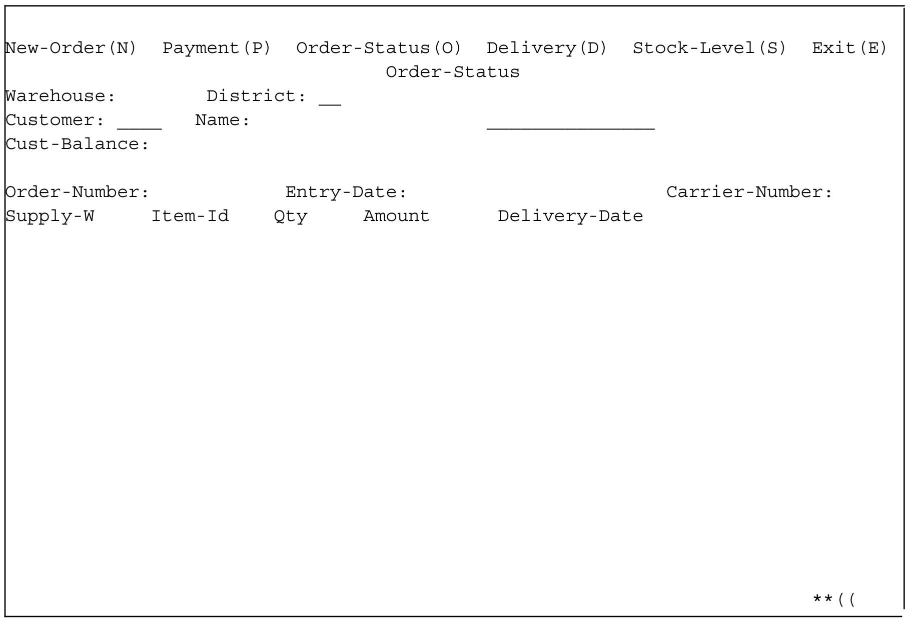
```
New-Order(N)   Payment(P)   Order-Status(O)   Delivery(D)   Stock-Level(S)   Exit(E)
                                      Delivery
Warehouse:

Carrier Number: __

Execution Status:













                                                                          **((
```



```
New-Order(N)   Payment(P)   Order-Status(O)   Delivery(D)   Stock-Level(S)   Exit(E)
                                   Stock-level
Warehouse:        District:

Stock level Threshold: __

Low Stock:















                                                                          **((
```



```
New-Order(N)  Payment(P)  Order-Status(O)  Delivery(D)  Stock-Level(S)  Exit(E)
                                     Payment
Date:

Warehouse:                                District: __




Customer: ____  Cust-Warehouse: ____  Cust-District: __
Name:                       _______________      Since:
                                                  Credit:
                                                  %Disc:
                                                  Phone:

Amount Paid:          _______      New Cust-Balance:
Credit Limit:

Cust-Data:



                                                                          **((
```



```
New-Order(N)  Payment(P)  Order-Status(O)  Delivery(D)  Stock-Level(S)  Exit(E)
                                  Order-Status
Warehouse:        District: __
Customer: ____     Name:                  _______________
Cust-Balance:

Order-Number:           Entry-Date:                        Carrier-Number:
Supply-W     Item-Id    Qty     Amount      Delivery-Date














                                                                         **((
```

Appendix F: Screen Layouts

F

This Appendix contains the screen form layouts for the 5 transactions.

```
New-Order(N)  Payment(P)  Order-Status(O)  Delivery(D)  Stock-Level(S)  Exit(E)
                                   New Order
Warehouse:        District: __                       Date:
Customer: ____   Name:                    Credit:      %Disc:
Order Number:           Number of Lines:           W_tax:         D_tax:

 Supp_W  Item_Id  Item Name                 Qty  Stock  B/G  Price    Amount
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
Execution Status:                                            Total:
                                                                        **((
                statsp->newo_think += (double) del;
                /* Save think time in histogram bucket */
        }
}
```

```
ol_cnt = random(5, 15);
rbk = random(1, 100);/* trans. to be rolledback */
sprintf(bufp, "%02d%04d", no.d_id, no.c_id);
bufp += strlen(bufp);
/* Generate all the item fields */
for (i=0; i < ol_cnt; i++, itemp++) {
        itemp->ol_i_id = NURand(8191, 1, 100000, CONST_IID);
        /* If last item and rbk, select unused item */
        if (i == ol_cnt - 1 && rbk == 1) {
                itemp->ol_i_id = 100001;
        }
        x = random(1, 100);
        if ( x > 1)
                itemp->ol_supply_w_id = W_ID;
        else {
                /* Select a warehouse other than w_id */
                do {
                        x = random(1, control->scale);
                } while (x == W_ID);
                itemp->ol_supply_w_id = x;
                remote++;
        }
        itemp->ol_quantity = random(1, 10);
        sprintf(bufp, "%04d%06d%02d",itemp->ol_supply_w_id,
        itemp->ol_i_id, itemp->ol_quantity);
        bufp += strlen(bufp);
}
strcpy(bufp, leave_key);
bufp += 2;
/* Compute keying time info */
end_time = gettime();
key_time = end_time - start_time;
start_time = end_time;

/* Now send fields to client */
/* Read output screen from client */
end_time = gettime();
/* Store elapse time info for thruput */
elapse_time = end_time - control->start_time;
/* compute the how long it took to run the tx */
resp_time = end_time - start_time + control->newo_delta;
/* Wait think time */
del = delay(control->newo_think, 5*control->newo_think);
poll(0, 0, del + control->newo_delta);
end_time = gettime();
/* Now post all stats */
if ( ! in_ramp && end_time <= control->end_stdystate) {
        statsp->newo_cnt++; /* another one bytes the dust */
        if ( rbk == 1 )
                statsp->newo_rbkcnt++;
        statsp->newo_remote += remote;
        statsp->newo_olcnt += ol_cnt;
        statsp->newo_key += key_time;
        /* Save keying time in histogram bucket */
        statsp->newo_resp += (double) resp_time;/* sum up the response time */
        /* Save response time in histogram bucket */
```

```
                if ( end_time >= control->end_rampup &&
                        end_time < control->end_stdystate )
                        in_ramp = 0;
                else
                        in_ramp = 1;
                if (end_time >= control->end_rampdown)
                        break;
        }
```

The do_menu function selects the transaction to execute based on the weighted distribution algorithm.

```
int
do_menu()
{
        int val, result, menu_start, menu_end, menu_resp;
        char ch;
        /* Read menu line from client */
        /* Choose tx. type*/
        /* Now select menu and compute menu response time */
        menu_start = gettime();
        /* Write menu selection to client */
        /* Read input form for this transaction type */
        menu_end = gettime();
        menu_resp = menu_end - menu_start;
        if ( ! in_ramp) {
                statsp->menu_resp += menu_resp;
                /* Post in histogram bucket */
                if ((menu_resp / MENU_BUCKET) < MENU_MAX)
                        statsp->menu_hist[menu_resp / MENU_BUCKET]++;
                else
                        statsp->menu_hist[MENU_MAX - 1]++;
                if (menu_resp > statsp->menu_max)
                        statsp->menu_max = menu_resp;
        }
        return(result);
}
/*
 * Function: do_neworder
 * This function executes the neworder transaction
 * It generates all the input fields, sends it to the
 * client over the keying time, measures the response
 * time, reads the results and delays for the think time.
 */
 /* The code for the other transactions is similar */
do_neworder()
{
        struct newo_fld no;
        struct items_fld *itemp = no.items;
        int ol_cnt, rbk, remote = 0, i, x;
        char *bufp = fldbuf;
        int start_time, end_time, key_time, resp_time, elapse_time, del;
        start_time = gettime();
        /* Now wait for keying time */
        poll (0, 0, NEWO_KEY);
        /* Generate all input data */
        no.d_id = random(1, 10);
        no.c_id = NURand(1023, 1, 3000, CONST_CID);
```

Appendix E: Driver Scripts

The following code sections show how the transactions are generated and how statistics are gathered. Each of the transaction functions generates the input data for that transaction, sends it to the client, reads the output form and computes keying, response and think time statistics.

This is the main loop of the RTE:

```
/* run for ramp up without capturing the stats */
        i=0;
        in_ramp = 1;
        while (1)
        {
                tx_type = do_menu(); /* Select transaction */
                switch (tx_type) {
                case NEWORDER:
                        do_neworder();
                        break;
                case PAYMENT:
                        do_payment();
                        break;
                case DELIVERY:
                        do_delivery();
                        break;
                case ORDSTAT:
                        do_ordstat();
                        break;
                case STOCKLEVEL:
                        do_stocklevel();
                        break;
                default:
                        fprintf(stderr, "%s: Slave %d: Internal error. Tx-type = %d\n",
                        hostname, slave_num, tx_type);
                        cleanup(-1);
                }
                end_time = gettime();
```

Warehouses	116.00		Logpage/Tx	1.81
tpmC	1,332.50			

Table	Rows	Initial Data	Population Index	(pages) Overhead
Customer	3480000	1160000	104569	306
District	1160	1160	7	22
History	3480000	97456	0	36
Item	100000	4762	43	11
Neworder	1044000	5,705.	74	10
Order	3480000	47,028.	611	9
Orderline	34800000	1,054,546.	14951	243
Stock	11600000	1,933,334.	23147	245
Warehouse	116	116	4	23
Total		4304107	143406	905

Segment Name	Physical Total Size	Storage Overhead
Scache	116,224.	3632
Sorders	71,680.	2240
Scustomer	1,244,160.	38880
Sorder_line	1,274,880.	39840
Shistory	163,840.	5120
Sstock	2,088,960.	65280
Scidx	97,280.	3040
Total	5,057,024.	158032

5% Growth	155,253.85		
Free space	295,320.15		
Dynamic Space	1,199,030.00		
Static Space	3,562,673.85		
Daily Growth	220,373.44		
Daily Spread	-35,240.02		
180day Space	43,229,893.92	In MB =	84,433.39
		In GB =	82.45
Logpages/Tx	1.81		
8hr LogSpace	1,157,676.00	In MB =	2,261.09
		In GB =	2.21
		Mirror (GB) =	2.21
Log+Mirror (GB)	4.42		

Appendix D: Disk Storage

The calculations used to determine the storage requirements for the 8 hour logical log and the 180-day space calculations are contained in this appendix.

The calculations for the 8 hour recovery log was determined as follows :

The number of logpages used during the measurement run was determined by using the Sybase stored procedure *sp_helpdb* tpcc before and after the run. The number of pages per new-order transaction was then computed. This was 1.81 pages. Using 1.81pages per transaction yields :

(2KB * 1.81 * 1332.50 * 60 * 8) / 1024 = 2261.09 MB.

We had allocated 2400 MB of log space.

```
newo_sybase SRVGRP=group1 SRVID=192  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=193  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=194  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=195  RQADDR=newoq1  REPLYQ=N

paym_sybase SRVGRP=group5 SRVID=21  RQADDR=paymq1  REPLYQ=N
paym_sybase SRVGRP=group5 SRVID=22  RQADDR=paymq1  REPLYQ=N
paym_sybase SRVGRP=group5 SRVID=23  RQADDR=paymq1  REPLYQ=N
paym_sybase SRVGRP=group5 SRVID=24  RQADDR=paymq1  REPLYQ=N
paym_sybase SRVGRP=group5 SRVID=25  RQADDR=paymq1  REPLYQ=N
paym_sybase SRVGRP=group5 SRVID=26  RQADDR=paymq1  REPLYQ=N
paym_sybase SRVGRP=group5 SRVID=27  RQADDR=paymq1  REPLYQ=N
paym_sybase SRVGRP=group5 SRVID=28  RQADDR=paymq1  REPLYQ=N
paym_sybase SRVGRP=group5 SRVID=29  RQADDR=paymq1  REPLYQ=N
paym_sybase SRVGRP=group5 SRVID=30  RQADDR=paymq1  REPLYQ=N
paym_sybase SRVGRP=group5 SRVID=301  RQADDR=paymq1  REPLYQ=N
paym_sybase SRVGRP=group5 SRVID=302  RQADDR=paymq1  REPLYQ=N
paym_sybase SRVGRP=group5 SRVID=303  RQADDR=paymq1  REPLYQ=N

*SERVICES
NEWO
PAYM
ORDS
DEL
STOCK
```

Compilation Flags

These are the compilation flags used to compile the application code:

```
-O -L/export/home/sybase/lib -lsybdb -lm -lc -lnsl
```

```
ords_sybase SRVGRP=group2 SRVID=34  RQADDR=ordsq1  REPLYQ=N
ords_sybase SRVGRP=group2 SRVID=35  RQADDR=ordsq1  REPLYQ=N
ords_sybase SRVGRP=group2 SRVID=36  RQADDR=ordsq1  REPLYQ=N
ords_sybase SRVGRP=group2 SRVID=37  RQADDR=ordsq1  REPLYQ=N
ords_sybase SRVGRP=group2 SRVID=38  RQADDR=ordsq1  REPLYQ=N
ords_sybase SRVGRP=group2 SRVID=39  RQADDR=ordsq1  REPLYQ=N
ords_sybase SRVGRP=group2 SRVID=40  RQADDR=ordsq1  REPLYQ=N
ords_sybase SRVGRP=group2 SRVID=401  RQADDR=ordsq1  REPLYQ=N

del_sybase SRVGRP=group4 SRVID=41  RQADDR=delq1  REPLYQ=N CLOPT="-A -- 1"
del_sybase SRVGRP=group4 SRVID=42  RQADDR=delq1  REPLYQ=N CLOPT="-A -- 2"
del_sybase SRVGRP=group4 SRVID=43  RQADDR=delq1  REPLYQ=N CLOPT="-A -- 3"
del_sybase SRVGRP=group4 SRVID=44  RQADDR=delq1  REPLYQ=N CLOPT="-A -- 4"
del_sybase SRVGRP=group4 SRVID=45  RQADDR=delq1  REPLYQ=N CLOPT="-A -- 5"
del_sybase SRVGRP=group4 SRVID=46  RQADDR=delq1  REPLYQ=N CLOPT="-A -- 6"
del_sybase SRVGRP=group4 SRVID=47  RQADDR=delq1  REPLYQ=N CLOPT="-A -- 7"
del_sybase SRVGRP=group4 SRVID=48  RQADDR=delq1  REPLYQ=N CLOPT="-A -- 8"

stock_sybase SRVGRP=group3 SRVID=51  RQADDR=stockq1  REPLYQ=N
stock_sybase SRVGRP=group3 SRVID=52  RQADDR=stockq1  REPLYQ=N
stock_sybase SRVGRP=group3 SRVID=53  RQADDR=stockq1  REPLYQ=N
stock_sybase SRVGRP=group3 SRVID=54  RQADDR=stockq1  REPLYQ=N
stock_sybase SRVGRP=group3 SRVID=55  RQADDR=stockq1  REPLYQ=N
stock_sybase SRVGRP=group3 SRVID=56  RQADDR=stockq1  REPLYQ=N
stock_sybase SRVGRP=group3 SRVID=57  RQADDR=stockq1  REPLYQ=N
stock_sybase SRVGRP=group3 SRVID=58  RQADDR=stockq1  REPLYQ=N
stock_sybase SRVGRP=group3 SRVID=59  RQADDR=stockq1  REPLYQ=N
stock_sybase SRVGRP=group3 SRVID=60  RQADDR=stockq1  REPLYQ=N
stock_sybase SRVGRP=group3 SRVID=61  RQADDR=stockq1  REPLYQ=N
stock_sybase SRVGRP=group3 SRVID=62  RQADDR=stockq1  REPLYQ=N
stock_sybase SRVGRP=group3 SRVID=63  RQADDR=stockq1  REPLYQ=N
stock_sybase SRVGRP=group3 SRVID=64  RQADDR=stockq1  REPLYQ=N
stock_sybase SRVGRP=group3 SRVID=65  RQADDR=stockq1  REPLYQ=N
stock_sybase SRVGRP=group3 SRVID=66  RQADDR=stockq1  REPLYQ=N
stock_sybase SRVGRP=group3 SRVID=67  RQADDR=stockq1  REPLYQ=N

newo_sybase SRVGRP=group1 SRVID=1  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=2  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=3  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=4  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=5  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=6  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=7  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=8  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=9  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=10  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=11  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=12  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=13  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=14  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=15  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=16  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=17  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=18  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=19  RQADDR=newoq1  REPLYQ=N
newo_sybase SRVGRP=group1 SRVID=191  RQADDR=newoq1  REPLYQ=N
```

```
[User Environment]
        number of user connections = 80
        stack size = DEFAULT
        stack guard size = DEFAULT
        systemwide password expiration = DEFAULT
        permission cache entries = DEFAULT
        user log cache size = 4096
        user log cache spinlock ratio = DEFAULT

[Lock Manager]
        number of locks = 20000
        deadlock checking period = DEFAULT
        freelock transfer block size = DEFAULT
        max engine freelocks = DEFAULT
        address lock spinlock ratio = DEFAULT
        page lock spinlock ratio = DEFAULT
        table lock spinlock ratio = DEFAULT
```

Tuxedo Configuration values

```
*RESOURCES
IPCKEY 40001
MASTER playwright
PERM 0666
MODEL SHM
LDBAL N
MAXACCESSERS 1310
MAXSERVERS 75
MAXSERVICES 110
SCANUNIT 35
SANITYSCAN 5
BLOCKTIME 5
BBLQUERY 60

*MACHINES
playwright LMID=playwright
 ROOTDIR="/export/home/tuxedo"
 APPDIR="/export/home/dbbench/tuxedo"
 TUXCONFIG="/export/home/dbbench/tuxedo/tuxconfig.playwright"
 ULOGPFX="/export/home/dbbench/tuxedo/ULOG"

*GROUPS
group1 LMID=playwright GRPNO=1
group2 LMID=playwright GRPNO=2
group3 LMID=playwright GRPNO=3
group4 LMID=playwright GRPNO=4
group5 LMID=playwright GRPNO=5

*SERVERS

ords_sybase SRVGRP=group2 SRVID=31  RQADDR=ordsq1  REPLYQ=N
ords_sybase SRVGRP=group2 SRVID=32  RQADDR=ordsq1  REPLYQ=N
ords_sybase SRVGRP=group2 SRVID=33  RQADDR=ordsq1  REPLYQ=N
```

```
[O/S Resources]
	o/s file descriptors = 1024
	o/s asynch i/o enabled = DEFAULT
	max async i/os per engine = 1024
	max async i/os per server = 1024

[Physical Resources]

[Physical Memory]
	total memory = 237000
	additional network memory = 737280
	lock shared memory = DEFAULT

[Processors]
	max online engines = DEFAULT
	min online engines = DEFAULT
	current number online engines = DEFAULT
	engine adjust interval = DEFAULT
	perform all disk i/o on engine 0 = DEFAULT
	run user tasks on engine 0 = DEFAULT

[SQL Server Administration]
	number of open objects = DEFAULT
	number of open databases = DEFAULT
	audit queue size = DEFAULT
	default database size = DEFAULT
	identity burning set factor = DEFAULT
	allow nested triggers = DEFAULT
	allow updates to system tables = 1
	print deadlock information = DEFAULT
	default fill factor percent = DEFAULT
	number of mailboxes = DEFAULT
	number of messages = DEFAULT
	number of alarms = DEFAULT
	number of pre-allocated extents = DEFAULT
	event buffers per engine = DEFAULT
	cpu accounting flush interval = DEFAULT
	i/o accounting flush interval = DEFAULT
	sql server clock tick length = DEFAULT
	runnable process search count = DEFAULT
	i/o polling process count = DEFAULT
	time slice = DEFAULT
	deadlock retries = DEFAULT
	cpu grace time = DEFAULT
	number of sort buffers = DEFAULT
	sort page count = DEFAULT
	number of extent i/o buffers = DEFAULT
	size of auto identity column = DEFAULT
	identity grab size = DEFAULT
	lock promotion HWM = DEFAULT
	lock promotion LWM = DEFAULT
	lock promotion PCT = DEFAULT
	housekeeper free write percent = DEFAULT
	partition groups = DEFAULT
	partition spinlock ratio = DEFAULT
```

```
set max_nprocs=2700
set pt_cnt=1350
set maxusers=256

set bufhwm = 1024
set maxusers=256
```

RDBMS Configuration values

```
##############################################################################
#
#           Configuration File for the Sybase SQL Server
#
#           Please read the System Administration Guide (SAG)
#           before changing any of the values in this file.
#
##############################################################################



[Configuration Options]

[General Information]

[Backup/Recovery]
    recovery interval in minutes = 3000
    print recovery information = DEFAULT
    tape retention in days = DEFAULT

[Cache Manager]
    number of oam trips = DEFAULT
    number of index trips = DEFAULT
    procedure cache percent = 5
    memory alignment boundary = DEFAULT

[Disk I/O]
    allow sql server async i/o = 1
    disk i/o structures = DEFAULT
    page utilization percent = DEFAULT
    number of devices = 200

[Network Communication]
    default network packet size = 4096
    max network packet size = 4096
    remote server pre-read packets = DEFAULT
    number of remote connections = 80
    allow remote access = DEFAULT
    number of remote logins = 20
    number of remote sites = DEFAULT
    max number network listeners = 20
    tcp no delay = DEFAULT
```

Appendix C: Tunable Parameters

This Appendix contains the configuration information for the operating system, the RDBMS and Tuxedo.

Operating System Configuration Values

The Solaris 2.5 kernel configuration parameters set in the file /etc/system are given below.

Solaris 2.5 Configuration File for Ultra Enterprise 150

```
set set shmsys:shminfo_shmmax=2147483647
set bufhwm = 4096
set maxusers=64
```

Solaris 2.5 configuration file for the client system:

```
set shmsys:shminfo_shmmax=268435456
set msgsys:msginfo_msgmax=32764
set msgsys:msginfo_msgmnb=16382
set msgsys:msginfo_msgmni=1300
set msgsys:msginfo_msgtql=1300
set msgsys:msginfo_msgseg=41600
set msgsys:msginfo_msgssz=128

set semsys:seminfo_semmap=40
set semsys:seminfo_semmsl=180
set semsys:seminfo_semmns=1340
set semsys:seminfo_semmnu=1340
set semsys:seminfo_semume=1340
```

```
#ifndef TPCC_INCLUDED
#define TPCC_INCLUDED

#include <sybfront.h>
#include <sybdb.h>
#include <time.h>


/* Population constants */
#define MAXITEMS 100000
#define CUST_PER_DIST 3000
#define DIST_PER_WARE 10
#define ORD_PER_DIST 3000

#defineNURAND_C 123

/* Types of application variables */
typedef int     COUNT;
typedef int     ID;
typedef double  MONEY;
typedef double  FLOAT;
typedef char    TEXT;
typedef struct { int x[2];} DATE;
typedef int     LOGICAL;

typedef enum
    {COUNT_T, ID_T, MONEY_T, FLOAT_T, TEXT_T,
     DATE_T, LOGICAL_T, MAX_T}
    DATA_TYPE;

typedef struct timeval TIME;


#define YES 1
#define NO 0
#define EOF (-1)

#ifndef NULL
#define NULL ((void *)0)
#endif

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

/* define function types */
externint   msg_handler();
externint   err_handler();
externint   batch_size;

#endif /* TPCC_INCLUDED */
```

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

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

        return length;
}

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

        length = 4;

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

        return length;
}


int MakeAlphaString(min, max, str)
        int min;
        int max;
        TEXT str[];
{
        static char character[] =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXY
Z1234567890";
        int length;
        int i;

        length = RandomNumber(min, max);

        for (i=0;  i<length;  i++)
        str[i] = character[RandomNumber(0, sizeof(character)-2)];
        str[length] = '\0';

        return length;
}


Original(str)
        TEXT str[];
{
        int pos;
        int len;

        len = strlen(str);
        if (len < 8) return;

        pos = RandomNumber(0,len-8);

        str[pos+0] = 'O';
        str[pos+1] = 'R';
        str[pos+2] = 'I';
        str[pos+3] = 'G';
        str[pos+4] = 'I';
        str[pos+5] = 'N';
        str[pos+6] = 'A';
        str[pos+7] = 'L';
}



RandomPermutation(perm, n)
        int perm[];
        int n;
{
        int i, r, t;

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

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


int Randomize()
{
        srand48(time(0)+getpid());
}


int RandomNumber(min, max)
        int min;
        int max;
{
        int r;
        r = (int)(drand48() * (max - min + 1)) + min;
        return r;
}


int NURandomNumber(a, c, min, max)
        int a;
        int c;
        int min;
        int max;
{
        int r;

        r = ((RandomNumber(0, a) | RandomNumber(min, max)) + c)
        % (max - min + 1) + min;

        return r;
}
*************************************************
```

```
        s_i_id, s_w_id,   s_data);
      bulk_load(bulk_s);
}


end_stock_load()
{
      bulk_close(bulk_s);
}



test(){}


getargs(argc, argv)

/*****************************************************************
configure configures the load stuff
  By default, loads all the tables for a the specified warehouse.
                When loading warehouse 1, also loads the item table.
******************************************************************/
      int argc;
      char **argv;
{
      char ch;

      /* define the defaults */
      load_item = load_warehouse = load_district = load_history =
      load_orders = load_new_order = load_order_line =
      load_customer = load_stock = NO;


      if        (strcmp(argv[1], "warehouse") == 0) load_warehouse =
YES;
      else if (strcmp(argv[1], "district") == 0)  load_district = YES;
      else if (strcmp(argv[1], "stock") == 0)  load_stock = YES;
      else if (strcmp(argv[1], "item") == 0)  load_item = YES;
      else if (strcmp(argv[1], "history") == 0)  load_history = YES;
      else if (strcmp(argv[1], "orders") == 0)  load_orders = YES;
      else if (strcmp(argv[1], "customer") == 0)  load_customer =
YES;
      else if (strcmp(argv[1], "new_order") ==0) load_new_order =
YES;
      else
      {
             printf("%s is not a valid table name\n", argv[1]);
             exit(0);

      }

      /* Set the w1 and w2 to argv[2] and arg[3] */
      if (argc < 3)
      {
            printf("Usage: %s <table> <w_first> [<w_last>]\n",
argv[0]);
            exit(1);
      }
      {
            w1 = atoi(argv[2]);
            if (argc >= 3)
                  w2 = atoi(argv[3]);
            else
                  w2 = w1;
      }

      /* Get the password for sa */
      if (argc > 4)
      strcpy(password,argv[4]);

      /* Check if warehouse is within the range */
      if (w1 <= 0 || w2 > 1000 || w1 > w2)
      {
            printf("Warehouse id is out of range\n");
            exit(0);
      }
}

double drand48();

MakeAddress(str1, str2, city, state, zip)
      TEXT str1[20+1];
      TEXT str2[20+1];
      TEXT city[20+1];
      TEXT state[2+1];
      TEXT zip[9+1];
{
      MakeAlphaString(10,20,str1);
      MakeAlphaString(10,20,str2);
      MakeAlphaString(10,20,city);
      MakeAlphaString(2,2,state);
      MakezipString(0,9999,zip);

      /* Changed for TPCC V 3.0 */
      strcat(zip, "11111");

}



LastName(num, name)
/*************************************************************
Lastname generates a lastname from a number.
*************************************************************/
      int num;
      char name[20+1];
{
      int i;
      static char *n[] = {"BAR", "OUGHT", "ABLE", "PRI", "PRES",
                                "ESE", "ANTI", "CALLY", "ATION", "EING"};

      strcpy(name, n[(num/100)%10]);
      strcat(name, n[(num/10) %10]);
      strcat(name, n[(num/1)  %10]);
}


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

```
TEXT s_dist_07[24+1];
TEXT s_dist_08[24+1];
TEXT s_dist_09[24+1];
TEXT s_dist_10[24+1];
COUNT s_ytd;
COUNT s_order_cnt;
COUNT s_remote_cnt;
TEXT s_data[50+1];

int bulk_s;

/*
** On loading stock in major order of item_id:
** 10% of the MAXITEMS items in each warehouse need to marked as
original
** (i.e., s_data like '%ORIGINAL%'.)  This is a bit harder to do when
we
** load by item number, rather than by warehouses.  The trick is to
first
** generate a huge WAREBATCH * MAXITEMS bitmap, initialize all
bits to zero,
** and then set 10% of bits in each row to 1.  While loading item i in
** warehouse w, we simply lookup bitmap[w][i] to see whether it
needs to
** be marked as original.
*/

LoadStock(w1, w2)
        ID w1, w2;
{
        ID w_id;

        BitVector original[WAREBATCH][((MAXITEMS+(WSZ-
1))/WSZ)], * bmp;
        int w, i, j;

        if (w2-w1+1 > WAREBATCH)
        {
                fprintf(stderr, "Can't load stock for %d warehouses.\n",
                        w2-w1+1);
                fprintf(stderr, "Please use batches of %d.\n",
WAREBATCH);
        }

        for (w=w1; w<=w2; w++)
        {
                bmp = original[w-w1];
                /* Mark all items as not "original" */
                for (i=0; i<(MAXITEMS+(WSZ-1))/WSZ; i++)
                        bmp[i] = (BitVector)0x0000;
                /* Mark exactly 10% of items as "original" */
                for (i=0; i<(MAXITEMS+9)/10; i++)
                {
                        do {
                                j = RandomNumber(0,MAXITEMS-1);
                        } while (nthbit(bmp,j));
                        setbit(bmp,j);
                }
        }

        printf("Loading stock for warehouse %d to %d.\n", w1, w2);
        begin_stock_load();
        /* do for each item */
        for (s_i_id=1;   s_i_id <= MAXITEMS; s_i_id++)
        {
                for (w_id=w1; w_id<=w2; w_id++)
                {
                /* Generate Stock Data */
                        s_w_id = w_id;
                        s_quantity = RandomNumber(10,100);
                        MakeAlphaString(24, 24, s_dist_01);
                        MakeAlphaString(24, 24, s_dist_02);
                        MakeAlphaString(24, 24, s_dist_03);
                        MakeAlphaString(24, 24, s_dist_04);
                        MakeAlphaString(24, 24, s_dist_05);
                        MakeAlphaString(24, 24, s_dist_06);
                        MakeAlphaString(24, 24, s_dist_07);
                        MakeAlphaString(24, 24, s_dist_08);
                        MakeAlphaString(24, 24, s_dist_09);
                        MakeAlphaString(24, 24, s_dist_10);
                        s_ytd = 0;
                        s_order_cnt = 0;
                        s_remote_cnt = 0;
                        MakeAlphaString(26, 50, s_data);
                        if (nthbit(original[w_id-w1],s_i_id-1))
                        {
                                Original(s_data);
                        }
                        stock_load();
                }
        }
        end_stock_load();
        printf("\nLoaded stock for warehouses %d to %d.\n", w1, w2);
}

begin_stock_load()
{
        int     i = 1;

        bulk_s = bulk_open("tpcc", "stock", password);

        bulk_bind(bulk_s, i++, "s_i_id", &s_i_id, ID_T);
        bulk_bind(bulk_s, i++, "s_w_id", &s_w_id, ID_T);
        bulk_bind(bulk_s, i++, "s_quantity", &s_quantity, COUNT_T);
        bulk_bind(bulk_s, i++, "s_ytd", &s_ytd,COUNT_T);
        bulk_bind(bulk_s, i++, "s_order_cnt", &s_order_cnt, COUNT_T);
        bulk_bind(bulk_s, i++, "s_remote_cnt", &s_remote_cnt, COUNT_T);
        bulk_bind(bulk_s, i++, "s_dist_01", s_dist_01, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_02", s_dist_02, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_03", s_dist_03, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_04", s_dist_04, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_05", s_dist_05, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_06", s_dist_06, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_07", s_dist_07, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_08", s_dist_08, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_09", s_dist_09, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_10", s_dist_10, TEXT_T);
        bulk_bind(bulk_s, i++, "s_data",s_data, TEXT_T);
}



stock_load()
{
        debug("s_i_id=%d w_id=%d s_data=%s\n",
        else            ol_amount = RandomNumber(1, 999999)  /  100.0;
        MakeAlphaString(24, 24, ol_dist_info);
        order_line_load();
}


NewOrder(w_id, d_id)
        ID w_id, d_id;
{

        no_d_id = o_d_id;
        no_w_id = o_w_id;
        for (no_o_id=2101;  no_o_id <= ORD_PER_DIST; no_o_id++)
        new_order_load();
}




begin_order_load()
{
        int     i = 1;

        o_bulk = bulk_open(“tpcc”, “orders”, password);

        bulk_bind(o_bulk, i++, “o_id”, &o_id, ID_T);
        bulk_bind(o_bulk, i++, “o_c_id”, &o_c_id, ID_T);
        bulk_bind(o_bulk, i++, “o_d_id”, &o_d_id, ID_T);
        bulk_bind(o_bulk, i++, “o_w_id”, &o_w_id, ID_T);
        bulk_bind(o_bulk, i++, “o_entry_d”, &o_entry_d, DATE_T);
        bulk_bind(o_bulk, i++, “o_carrier_id”, &o_carrier_id, ID_T);
        bulk_bind(o_ol_cnt, i++, “o_ol_cnt”, &o_ol_cnt, COUNT_T);
        bulk_bind(o_all_local, i++, “o_all_local”, &o_all_local,
LOGICAL_T);
}

order_load()
{
        debug(“o_id=%d o_c_id=%d  count=%d\n”, o_id, o_c_id,
o_ol_cnt);
        bulk_load(o_bulk);
}

end_order_load()
{
        bulk_close(o_bulk);
}



begin_order_line_load()
{
        int     i = 1;

        ol_bulk = bulk_open(“tpcc”, “order_line”, password);

        bulk_bind(ol_bulk, i++, “ol_o_id”, &ol_o_id, ID_T);
        bulk_bind(ol_bulk, i++, “ol_d_id”, &ol_d_id, ID_T);
        bulk_bind(ol_bulk, i++, “ol_w_id”, &ol_w_id, ID_T);
        bulk_bind(ol_bulk, i++, “ol_number”, &ol_number, ID_T);
        bulk_bind(ol_bulk, i++, “ol_i_id”, &ol_i_id, ID_T);
        bulk_bind(ol_bulk, i++, “ol_supply_w_id”, &ol_supply_w_id, ID_T);
        bulk_bind(ol_bulk, i++, “ol_delivery_d”, &ol_delivery_d, DATE_T);
        bulk_bind(ol_bulk, i++, “ol_quantity”, &ol_quantity, COUNT_T);
        bulk_bind(ol_bulk, i++, “ol_amount”, &ol_amount, MONEY_T);
        bulk_bind(ol_bulk, i++, “ol_dist_info”, ol_dist_info, TEXT_T);
}

order_line_load()
{
        static int ol_count = 0;
        debug(“  ol_o_id=%d  ol_number=%d  ol_amount=%g\n”,
                ol_o_id,ol_number,ol_amount);
        bulk_load(ol_bulk);
}


end_order_line_load()
{
        bulk_close(ol_bulk);
}

begin_new_order_load()
{
        int     i = 1;

        no_bulk = bulk_open(“tpcc”, “new_order”, password);

        bulk_bind(no_bulk, i++, “no_o_id”, &no_o_id, ID_T);
        bulk_bind(no_bulk, i++, “no_d_id”, &no_d_id, ID_T);
        bulk_bind(no_bulk, i++, “no_w_id”, &no_w_id, ID_T);
}

new_order_load()
{
        debug(“  no_o_id=%d   \n”, no_o_id);
        bulk_load(no_bulk);
}


end_new_order_load()
{
        bulk_close(no_bulk);
}


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

Stock

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


ID s_i_id;
ID s_w_id;
COUNT s_quantity;
TEXT s_dist_01[24+1];
TEXT s_dist_02[24+1];
TEXT s_dist_03[24+1];
TEXT s_dist_04[24+1];
TEXT s_dist_05[24+1];
TEXT s_dist_06[24+1];
```

```
ID no_d_id;
ID no_w_id;


int  o_bulk;
int ol_bulk;
int no_bulk;



LoadOrd(w1, w2)
        ID w1, w2;
{
        ID w_id;
        ID d_id;

        begin_order_load();
        begin_order_line_load();
        for (w_id=w1; w_id<=w2; w_id++)
        {

                for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++)
                        Orders(w_id, d_id);

                printf("\nLoaded order + order_line for warehouse
%d\n", w_id);
        }
        end_order_line_load();
        end_order_load();
}

LoadNew(w1, w2)
        ID w1, w2;
{
        ID w_id;
        ID d_id;

        begin_new_order_load();
        for (w_id=w1; w_id<=w2; w_id++)
        {
                for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++)
                {
                        no_d_id = d_id;
                        no_w_id = w_id;
                        for (no_o_id=2101;  no_o_id <= ORD_PER_DIST;
no_o_id++)
                                new_order_load();
                }
                printf("\nLoaded new_order for warehouse %d\n",
w_id);
        }
        end_new_order_load();
}

Orders(w_id, d_id)
        ID w_id;
        ID d_id;
{
        int cust[ORD_PER_DIST+1];
        int ol_cnt[ORD_PER_DIST+1], sum;
        ID ol;

        printf("\nLoading orders and order lines  for warehouse %d district %d\n",
         w_id, d_id);

        RandomPermutation(cust, ORD_PER_DIST);


        for (o_id = 1, sum=0;  o_id <= ORD_PER_DIST;  o_id++)
                sum += (ol_cnt[o_id] = RandomNumber(5, 15));

        while (sum > 10*ORD_PER_DIST)
        {
                do {
                        o_id = RandomNumber(1,ORD_PER_DIST);
                } while (ol_cnt[o_id]==5);
                ol_cnt[o_id]--;
                sum--;
        }

        while (sum < 10*ORD_PER_DIST)
        {
                do {
                        o_id = RandomNumber(1,ORD_PER_DIST);
                } while (ol_cnt[o_id]==15);
                ol_cnt[o_id]++;
                sum++;
        }

        for (o_id = 1;  o_id <= ORD_PER_DIST;  o_id++)
        {
        o_c_id = cust[o_id];
        o_d_id = d_id;
        o_w_id = w_id;
        datetime(&o_entry_d);
        if (o_id <= 2100)
        o_carrier_id = RandomNumber(1,10);
        else o_carrier_id = -1;
        o_ol_cnt = ol_cnt[o_id];
         /* o_ol_cnt = RandomNumber(5, 15); */
        o_all_local = 1;
        order_load();

        for (ol=1;  ol<=o_ol_cnt; ol++)
                OrderLine(ol);

        }
}



OrderLine(ol)
        ID ol;
{

        ol_o_id = o_id;
        ol_d_id = o_d_id;
        ol_w_id = o_w_id;
        ol_number = ol;
        ol_i_id = RandomNumber(1, MAXITEMS);
        ol_supply_w_id = o_w_id;
        ol_delivery_d = o_entry_d;
        ol_quantity = 5;
        if (o_id <= 2100) ol_amount = 0;
```

```
                }
        }

        c_w_id = w_id;
        for (i=0; i<CUST_PER_DIST; i++)
        {
                c_id = i+1;
                for (d_id=1;  d_id <= DIST_PER_WARE;  d_id++)
                {
                        c_d_id = d_id;

LastName(i<1000?i:NURandomNumber(255,NURAND_C,0,999),c_la
st);
                        MakeAlphaString(8, 16, c_first);

MakeAddress(c_street_1,c_street_2,c_city,c_state,c_zip);
                        MakeNumberString(16, 16, c_phone);
                        MakeAlphaString(300, 500, c_data);
                        datetime(&c_since);
                        c_credit[0] = nthbit(badcredit[d_id-1],i) ? ‘B’ : ‘G’;
                        c_discount = RandomNumber(0, 50)  /  100.0;
                        c_balance = -10.0;
                        customer_load();
                }
        }
}

begin_customer_load()
{
        int     i = 1;

        bulk_c = bulk_open(“tpcc”, “customer”, password);

        bulk_bind(bulk_c, i++, “c_id”,  &c_id,   ID_T);
        bulk_bind(bulk_c, i++, “c_d_id”,&c_d_id, ID_T);
        bulk_bind(bulk_c, i++, “c_w_id”,&c_w_id, ID_T);
        bulk_bind(bulk_c, i++, “c_first”,   c_first, TEXT_T);
        bulk_bind(bulk_c, i++, “c_middle”,  c_middle, TEXT_T);
        bulk_bind(bulk_c, i++, “c_last”,c_last,    TEXT_T);
        bulk_bind(bulk_c, i++, “street_1”,  c_street_1 , TEXT_T);
        bulk_bind(bulk_c, i++, “street_2”,  c_street_2,  TEXT_T);
        bulk_bind(bulk_c, i++, “c_city”,c_city,  TEXT_T);
        bulk_bind(bulk_c, i++, “c_state”,   c_state, TEXT_T);
        bulk_bind(bulk_c, i++, “c_zip”, c_zip,   TEXT_T);
        bulk_bind(bulk_c, i++, “c_phone”,   c_phone, TEXT_T);
        bulk_bind(bulk_c, i++, “c_since”,   &c_since, DATE_T);
        bulk_bind(bulk_c, i++, “c_credit”,  c_credit,TEXT_T);
        bulk_bind(bulk_c, i++, “c_credit_lim”, &c_credit_lim,
MONEY_T);
        bulk_bind(bulk_c, i++, “c_discount”, &c_discount, FLOAT_T);
        bulk_bind(bulk_c, i++, “c_delivery_cnt”, &c_delivery_cnt,
COUNT_T);
        bulk_bind(bulk_c, i++, “c_payment_cnt”,&c_payment_cnt,
COUNT_T);
        bulk_bind(bulk_c, i++, “c_balance”, &c_balance, MONEY_T);
        bulk_bind(bulk_c, i++, “c_ytd_payment”, &c_ytd_payment,
MONEY_T);
        bulk_bind(bulk_c, i++, “c_data_1”, c_data1, TEXT_T);
        bulk_bind(bulk_c, i++, “c_data_2”, c_data2, TEXT_T);
}

customer_load()
{
        debug(“c_id=%-5d  d_id=%-5d  w_id=%-5d  c_last=%s\n”,
          c_id,c_d_id,  c_w_id,  c_last);


        /* Break the string c_data into 2 pieces */
        len = strlen(c_data);
        if (len > 250)
                {
                memcpy(c_data1, c_data, 250);
                c_data1[250]=’\0’;
                memcpy(c_data2, c_data+250, len-250 +1);
                }

        else
                {
                memcpy(c_data1, c_data, 250+1);
                strcpy(c_data2,””);
                }

        /* load the data */
        bulk_load(bulk_c);
}

end_customer_load()
{
        bulk_close(bulk_c);
}



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

Order, Order line, New order

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

/* Order row */
ID o_id;
ID o_c_id;
ID o_d_id;
ID o_w_id;
DATE o_entry_d;
ID o_carrier_id;
COUNT o_ol_cnt;
LOGICAL o_all_local;

/* Order line row */
ID   ol_o_id;
ID   ol_d_id;
ID   ol_w_id;
ID   ol_number;
ID   ol_i_id;
ID   ol_supply_w_id;
DATE ol_delivery_d;
COUNT ol_quantity;
MONEY ol_amount;
TEXT  ol_dist_info[24+1];

/* new order row */
ID no_o_id;
```

```
        printf("\nLoaded history for warehouse %d\n", w_id);
    }
    end_history_load();
}


LoadCustHist(w_id, d_id, c_id)
    ID w_id, d_id, c_id;
{

    h_c_id = c_id;
    h_c_d_id = d_id;
    h_c_w_id = w_id;
    h_d_id = d_id;
    h_w_id = w_id;
    h_amount = 10.0;
    MakeAlphaString(12, 24,h_data);
    datetime(&h_date);
    history_load();
}



begin_history_load()
{
    int    i = 1;

    bulk_h = bulk_open("tpcc", "history", password);

    bulk_bind(bulk_h, i++, "h_c_id",   &h_c_id,   ID_T);
    bulk_bind(bulk_h, i++, "h_c_d_id", &h_c_d_id, ID_T);
    bulk_bind(bulk_h, i++, "h_c_w_id", &h_c_w_id, ID_T);
    bulk_bind(bulk_h, i++, "h_d_id",   &h_d_id,   ID_T);
    bulk_bind(bulk_h, i++, "h_w_id",   &h_w_id,   ID_T);
    bulk_bind(bulk_h, i++, "h_date",   &h_date,   DATE_T);
    bulk_bind(bulk_h, i++, "h_amount", &h_amount, MONEY_T);
    bulk_bind(bulk_h, i++, "h_data",   h_data,    TEXT_T);
}

history_load()
{

    debug("h_c_id=%d  h_amount=%g\n", h_c_id, h_amount);
    bulk_load(bulk_h);
}


end_history_load()
{
    bulk_close(bulk_h);
}



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

Customer

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

/* static variables containing fields for customer record */
ID c_id;
ID c_d_id;
ID c_w_id;
TEXT c_first[16+1];
TEXT c_middle[2+1] = "OE";
TEXT c_last[16+1];
TEXT c_street_1[20+1];
TEXT c_street_2[20+1];
TEXT c_city[20+1];
TEXT c_state[2+1];
TEXT c_zip[9+1];
TEXT c_phone[16+1];
DATE c_since;
TEXT c_credit[2+1] = "?C";
MONEY c_credit_lim = 50000.0;
FLOAT c_discount;
MONEY c_balance = -10.0;
MONEY c_ytd_payment = 10.0;
COUNT c_payment_cnt = 1;
COUNT c_delivery_cnt = 0;
TEXT c_data[500+1];
TEXT c_data1[250+1];
TEXT c_data2[250+1];
ID    len;

int bulk_c;


LoadCustomer(w1, w2)
    ID w1, w2;
{
    ID w_id;

    begin_customer_load();
    for (w_id=w1; w_id<=w2; w_id++)
    {
        Customer(w_id);
        printf("\nLoaded customer for warehouse %d\n", w_id);
    }
    end_customer_load();
}

Customer(w_id)
    int w_id;
{
    BitVector badcredit[DIST_PER_WARE][(3000+WSZ-1)/WSZ], * bmp;
    int i, j;
    ID d_id;

    /* Mark exactly 10% of customers as having bad credit */
    for (d_id=1;  d_id <= DIST_PER_WARE;  d_id++)
    {
        bmp = badcredit[d_id-1];
        for (i=0; i<(3000+WSZ-1)/WSZ; i++)
            bmp[i] = (BitVector)0x0000;
        for (i=0; i<(3000+9)/10; i++)
        {
            do {
                j = RandomNumber(0,3000-1);
            } while (nthbit(bmp,j));
            setbit(bmp,j);
```

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

Item

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



ID i_id;
ID i_im_id;
TEXT i_name[24+1];
MONEY i_price;
TEXT  i_data[50+1];

int bulk_i;

LoadItems()
{

        int perm[MAXITEMS+1];
        int i, r, t;

        printf("Loading items\n");

        begin_item_load();

        /* select exactly 10% of items to be labeled "original" */
        RandomPermutation(perm, MAXITEMS);

        /* do for each item */
        for (i_id=1;   i_id <= MAXITEMS; i_id++)
        {

        /* Generate Item Data */
        MakeAlphaString(14, 24, i_name);
        i_price = RandomNumber(100,10000)  /  100.0;
        MakeAlphaString(26, 50, i_data);
        if (perm[i_id] <= (MAXITEMS+9)/10)
                Original(i_data);

        /* Generate i_im_id for V 3.0 */
        i_im_id = RandomNumber(1, 10000);

         item_load();
         }

        end_item_load();
        return;
}



begin_item_load()
{
        int     i = 1;

        bulk_i = bulk_open("tpcc", "item", password);

        /* bind the variables to the sybase columns */
        bulk_bind(bulk_i, i++, "i_id", &i_id, ID_T);
        bulk_bind(bulk_i, i++, "i_im_id", &i_im_id, ID_T);
        bulk_bind(bulk_i, i++, "i_name", i_name, TEXT_T);
        bulk_bind(bulk_i, i++, "i_price", &i_price, MONEY_T);
        bulk_bind(bulk_i, i++, "i_data", i_data, TEXT_T);
}



item_load()
{
        debug("i_id=%3d price=%5.2f data=%s\n",
          i_id, i_price, i_data);
        bulk_load(bulk_i);
}


end_item_load()
{
        bulk_close(bulk_i);
}



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

History

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



ID h_c_id;
ID h_c_d_id;
ID h_c_w_id;
ID h_d_id;
ID h_w_id;
DATE h_date;
MONEY h_amount;
TEXT h_data[24+1];

int bulk_h;

LoadHist(w1, w2)
        ID w1, w2;
{
        ID w_id;
        ID d_id, c_id;

        begin_history_load();
        for (w_id=w1; w_id<=w2; w_id++)
        {

                for (d_id=1;  d_id <= DIST_PER_WARE; d_id++)
                {
                        for (c_id=1; c_id <= CUST_PER_DIST; c_id++)
                                LoadCustHist(w_id, d_id, c_id);
                }
        return;
}


begin_warehouse_load()
{
        int     i = 1;

        bulk_w = bulk_open("tpcc", "warehouse", password);

        bulk_bind(bulk_w, i++, "w_id",   &w_id, ID_T);
        bulk_bind(bulk_w, i++, "w_name", w_name, TEXT_T);
        bulk_bind(bulk_w, i++, "w_street_1", w_street_1, TEXT_T);
        bulk_bind(bulk_w, i++, "w_street_2", w_street_2, TEXT_T);
        bulk_bind(bulk_w, i++, "w_city", w_city, TEXT_T);
        bulk_bind(bulk_w, i++, "w_state", w_state, TEXT_T);
        bulk_bind(bulk_w, i++, "w_zip", w_zip, TEXT_T);
        bulk_bind(bulk_w, i++, "w_tax", &w_tax, FLOAT_T);
        bulk_bind(bulk_w, i++, "w_ytd", &w_ytd, MONEY_T);
}


warehouse_load()
{
        debug("Loading Warehouse %d\n", w_id);
        bulk_load(bulk_w);
}


end_warehouse_load()
{
        bulk_close(bulk_w);
}



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

District

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



ID d_id;
ID d_w_id;
TEXT d_name[10+1];
TEXT d_street_1[20+1];
TEXT d_street_2[20+1];
TEXT d_city[20+1];
TEXT d_state[2+1];
TEXT d_zip[9+1];
FLOAT d_tax;
MONEY d_ytd;
ID   d_next_o_id;

int bulk_d;

LoadDistrict(w1, w2)
        ID w1, w2;
{
        ID w_id;

        begin_district_load();
        for (w_id=w1; w_id<=w2; w_id++)
        {
                printf("Loading districts for warehouse %d\n", w_id);

                d_w_id = w_id;
                d_ytd = 30000.00;
                d_next_o_id = 3001;

                for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++)
                {

                        MakeAlphaString(6, 10, d_name);
                        MakeAddress(d_street_1, d_street_2, d_city, d_state, d_zip);
                        d_tax = RandomNumber(10,20) / 100.0;

                        district_load();
                }
                printf("loaded district for warehouse %d\n", w_id);
        }
        end_district_load();
        return;
}



begin_district_load()
{
        int     i = 1;

        bulk_d = bulk_open("tpcc", "district", password);

        bulk_bind(bulk_d, i++, "d_id", &d_id, ID_T);
        bulk_bind(bulk_d, i++, "d_w_id", &d_w_id, ID_T);
        bulk_bind(bulk_d, i++, "d_name", d_name, TEXT_T);
        bulk_bind(bulk_d, i++, "d_street_1", d_street_1, TEXT_T);
        bulk_bind(bulk_d, i++, "d_street_2", d_street_2, TEXT_T);
        bulk_bind(bulk_d, i++, "d_city", d_city, TEXT_T);
        bulk_bind(bulk_d, i++, "d_state", d_state, TEXT_T);
        bulk_bind(bulk_d, i++, "d_zip", d_zip, TEXT_T);
        bulk_bind(bulk_d, i++, "d_tax", &d_tax, FLOAT_T);
        bulk_bind(bulk_d, i++, "d_ytd", &d_ytd, MONEY_T);
        bulk_bind(bulk_d, i++, "d_next_o_id", &d_next_o_id, ID_T);
}

district_load()
{
        debug("District %d  w_id=%d\n", d_id, d_w_id);
        bulk_load(bulk_d);
}

end_district_load()
{
        bulk_close(bulk_d);
}
```

```
char            *servername;
char            *procname;
int             line;
{

        /* changing database messages */
        if (msgno == DUMB_MESSAGE || msgno == ABORT_ERROR
|| msgno == 5703 || msgno == 5704)
                return(SUCCEED);

        /* Is this a deadlock message */
        if (msgno == 1205)
        {
                /* Set the deadlock indicator */
                *((DBBOOL *) dbgetuserdata(dbproc)) = TRUE;

                /* Sleep a few seconds before going back */
                sleep((unsigned) 2);
                return(SUCCEED);

        }

        fprintf(stderr, "msg no %d -\n%s", msgno, msgtext);

        /* exit on any error */
        exit(-101);
}
*************************************************
typedefunsigned longBitVector;
#define WSZ(sizeof(BitVector)*8)
#ifndef WAREBATCH
#defineWAREBATCH200
#endif
#definenthbit(map,n)map[(n)/WSZ] &  (((BitVector)0x1)<<
((n)%WSZ))
#definesetbit(map,n)map[(n)/WSZ] |= (((BitVector)0x1)<<
((n)%WSZ))

/***********************************************************
Load TPCC tables
***********************************************************/
#include "stdio.h"
#include "string.h"
#include "loader.h"

int load_item;
int load_warehouse;
int load_district;
int load_history;
int load_orders;
int load_new_order;
int load_order_line;
int load_customer;
int load_stock;
ID  w1, w2;
ID  warehouse;
int batch_size = 1000;
char   password[10];

int main(argn, argv)
        int argn;
        char **argv;
{

        /* Setup to use  the dblib version 10 for numeric datatypes */
        dbsetversion(DBVERSION_100);

        getargs(argn, argv);
        Randomize();

        if (load_item)LoadItems();
        if (load_warehouse)LoadWarehouse(w1, w2);
        if (load_district)LoadDistrict(w1, w2);
        if (load_history)LoadHist(w1, w2);
        if (load_customer)LoadCustomer(w1, w2);
        if (load_stock)LoadStock(w1, w2);
        if (load_orders)LoadOrd(w1, w2);
        if (load_new_order)LoadNew(w1, w2);
        return 0;
}


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

Warehouse

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


ID   w_id;
TEXT w_name[10+1];
TEXT w_street_1[20+1];
TEXT w_street_2[20+1];
TEXT w_city[20+1];
TEXT w_state[2+1];
TEXT w_zip[9+1];
FLOAT w_tax;
MONEY w_ytd;


int bulk_w;

LoadWarehouse(w1, w2)
        ID w1, w2;
{

        begin_warehouse_load();
        for (warehouse=w1; warehouse<=w2; warehouse++)
        {
                printf("Loading warehouse for warehouse %d\n", warehouse);

                w_id = warehouse;
                MakeAlphaString(6, 10, w_name);
                MakeAddress(w_street_1, w_street_2, w_city, w_state, w_zip);

                w_tax = RandomNumber(10, 20) / 100.0;
                w_ytd = 300000.00;

                warehouse_load();

                printf("loaded warehouse for warehouse %d\n", warehouse);
        }
        end_warehouse_load();
        /* release the login record */
        dbloginfree(login);

        /* prepare to do a bulk copy */
        if (bcp_init(dbproc[db], table, NULL, NULL, DB_IN) !=
SUCCEED)
        printf("Can't initialize the bulk copy to table %s\n", table);

        return db;
}



bulk_bind(db, column, name, address, type)
        int db;
        int column;
        char name[];
        void *address;
        int type;
{
        if (bcp_bind(dbproc[db], address, 0, -1, parm[type].terminator,
                        parm[type].termlen, parm[type].type, column) !=
SUCCEED)
                printf("Can't bind column %d to 0x%x, type=%d\n",
                        column,address,type);
}

bulk_null(db, column)
        int db;
        int column;
{
        if (bcp_collen(dbproc[db], 0, column) != SUCCEED)
                printf("Can't null column %d\n", column);
}


bulk_non_null(db, column)
        int db;
        int column;
{
        if (bcp_collen(dbproc[db], -1, column) != SUCCEED)
                printf("Can't non-null column %d\n", column);
}


bulk_load(db)
        int db;
{
        count[db]++;
        if (bcp_sendrow(dbproc[db]) != SUCCEED)
                printf("bulk_load: Can't load row\n");
        if (count[db]%batch_size == 0 && (bcp_batch(dbproc[db]) == -
1))
        printf("bulk_load: Can't post rows\n");
        if (count[db]%1000 == 0) write(1,".",1);
        if (count[db]%50000 == 0) write(1,"\n",1);

}

bulk_close(db)
        int db;
{
        if (bcp_done(dbproc[db]) == -1)
                printf("Problems completing the bulk copy.\n");
        dbproc[db] = NULL;
        if (count[db] >= 1000) write(1,"\n",1);
}
*************************************************
#if ! lint
static char *sddsId = "@(#)  error.c  1.1  4/30/91  19:47:32";
#endif /* ! lint */

/*
** Confidential property of Sybase, Inc.
** (c) Copyright Sybase, Inc. 1991
** All rights reserved
*/

/*
** error.c:1.14/30/9119:47:32
**      Standard error handler for RungenII and supporting code
**
**      HMS [04/30/91]
*/

/* Required standard include files */
#include <stdio.h>

/* Required Sybase include files */
#include <sybfront.h>
#include <sybdb.h>

/* message numbers that we don't want to deal with */
#defineDUMB_MESSAGE5701
#defineABORT_ERROR6104

int
err_handler(dbproc, severity, errno, oserr)
  DBPROCESS *dbproc;
  int severity;
  int errno;
  int oserr;
{
        /* changing databases message */
        if (errno == DUMB_MESSAGE || errno == ABORT_ERROR)
                return(INT_CANCEL);

        fprintf(stderr,"DB-LIBRARY Error: \n\t%s\n",dberrstr(errno));

        if (oserr != DBNOERR)
                fprintf(stderr,"O/S Error: \n\t%s\n",dboserrstr(oserr));

        /* exit on any error */
        exit(-100);
}



int
msg_handler(dbproc,msgno,msgstate,severity,msgtext,servername,procname,line)
DBPROCESS*dbproc;
int             msgno;
int             msgstate;
int             severity;
char            *msgtext;
```

```
go

ALTER DATABASE tpcc log on tpcc_log2=800
go

DISK INIT name = “tpcc_log3”, physname = “/dev/TClog3”,
       vdevno = 49, size = 409600
go

ALTER DATABASE tpcc log on tpcc_log3=800
go

EOF
************************************************
#!/bin/csh

isql -e -Usa -P << EOF

DISK MIRROR name = “tpcc_log”, mirror = “/dev/TCmirror1”,
                   writes = noserial
go

DISK MIRROR name = “tpcc_log2”, mirror = “/dev/TCmirror2”,
                   writes = noserial
go

DISK MIRROR name = “tpcc_log3”, mirror = “/dev/TCmirror3”,
                   writes = noserial
go

EOF
************************************************
/*****************************************************************
******************************************************************

Sybase Specific Routines

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

#include <stdio.h>
#include <sys/time.h>
#include <string.h>
#include “loader.h”

datetime(date)
       DBDATETIME *date;
{
       struct timeval time;
       gettimeofday(&time, NULL);
       date->dtdays = time.tv_sec / (60*60*24)
                   + (1970-1900)*365 + (1970-1900)/4;
       date->dttime = (time.tv_sec % (60*60*24))*300
                   + time.tv_usec*300/1000000;
}


/* define the type information for each field */
typedef struct
{
       char *terminator;
       int termlen;
       int type;
} bind_parm;

bind_parm parm[MAX_T] =
{
       /* COUNT */{NULL, 0, SYBINT4},
       /* ID*/    {NULL, 0, SYBINT4},
       /* MONEY */{NULL, 0, SYBFLT8},
       /* FLOAT */{NULL, 0, SYBFLT8},
       /* TEXT */{““,   1, SYBCHAR},
       /* DATE */{NULL, 0, SYBDATETIME},
       /* LOGICAL */{NULL, 0, SYBINT4}
};

#define MAXOPENS 10

DBPROCESS *dbproc[MAXOPENS];
int    count[MAXOPENS];

int bulk_open(database, table, password)
       char database[];
       char table[];
       char password[];
{
       LOGINREC *login;
       int db;

       /* make note we have established a connection */
       for (db=0; db<MAXOPENS; db++)
              if (dbproc[db] == NULL) break;
       count[db] = 0;

       /* Install an error and Message handler */
       dbmsghandle(msg_handler);
       dberrhandle(err_handler);

       /* initialize dblib */
       if (dbinit() != SUCCEED)
              printf(“Can’t initialize the DB library\n”);

       /* allocate a login record and fill it in */
       login = dblogin();
       if (login == NULL)
              printf(“Can’t allocate a login record.\n”);
       DBSETLUSER(login, “sa”);

       if(strlen(password) > 0)
       DBSETLPWD(login, password);

       DBSETLAPP(login, table);
       BCP_SETL(login, TRUE);

       /* Set Packet Size to 4096 */
       DBSETLPACKET(login, 4096);

       /* establish a connection with the server specified by DSQUERY */
       dbproc[db] = dbopen(login, NULL);
       if (dbproc[db] == NULL)
              printf(“Can’t establish connection. Is DSQUERY set?\n”);

       /* select the database to use */
       if (database != NULL)
              if (dbuse(dbproc[db], database) != SUCCEED)
                     printf(“Can’t select database: %s\n”, database);
```

```
go
dbcc tune(oamtrips, 100, orders)
go

if exists ( select name from sysobjects where name = 'order_line' )
        drop table order_line
go
create table order_line (
        ol_o_id      int,
        ol_d_id      tinyint,
        ol_w_id      smallint,
        ol_numbertinyint,
        ol_i_id      int,
        ol_supply_w_idsmallint,
        ol_delivery_ddatetime,/*- Updated by D */
        ol_quantitysmallint,
        ol_amountfloat,
        ol_dist_infochar(24)
) on Sorder_line
go
create unique clustered index ol_clu
        on order_line(ol_w_id, ol_d_id, ol_o_id, ol_number)
        on Sorder_line
go
dbcc tune(ascinserts, 1, order_line)
go
dbcc tune(oamtrips, 100, order_line)
go

if exists ( select name from sysobjects where name = 'item' )
        drop table item
go
create table item (
        i_id         int,
        i_im_id      int,
        i_name       char(24),
        i_price      float,
        i_data       char(50)
) on Scache
go
create unique clustered index i_clu
        on item(i_id)
        on Scache
go
dbcc tune(indextrips, 10, item)
go

if exists ( select name from sysobjects where name = 'stock' )
        drop table stock
go
create table stock (
        s_i_id       int,
        s_w_id       smallint,
        s_quantitysmallint,/*- Updated by NO */
        s_ytd        int,            /*- Updated by NO */
        s_order_cntsmallint,/*- Updated by NO */
        s_remote_cntsmallint,/*- Updated by NO */
        s_dist_01char(24),
        s_dist_02char(24),
        s_dist_03char(24),
        s_dist_04char(24),
        s_dist_05char(24),
        s_dist_06char(24),
        s_dist_07char(24),
        s_dist_08char(24),
        s_dist_09char(24),
        s_dist_10char(24),
        s_data       char(50)
) on Sstock
go
create unique clustered index s_clu
        on stock(s_i_id, s_w_id)
        on Sstock
go
dbcc tune(indextrips, 10, stock)
go

checkpoint
go
EOF
************************************************
#!/bin/sh -f

isql -e -Usa -P$PASSWORD << EOF
/* This script will create the TPC-C indexes that are best
   created after the load. */
use tpcc
go

create unique clustered index w_clu
        on warehouse(w_id)
        with fillfactor = 1
        on Scache
go
dbcc tune(indextrips, 100, warehouse)
go

create unique clustered index d_clu
        on district(d_w_id, d_id)
        with fillfactor = 1
        on Scache
go
dbcc tune(indextrips, 100, district)
go

select getdate()
go
create unique nonclustered index c_non1
        on customer(c_w_id, c_d_id, c_last, c_first, c_id)
        on Scidx
go
select getdate()
go

checkpoint
go
EOF
************************************************
#!/bin/csh

isql -e -Usa -P << EOF
use master
go

DISK INIT name = "tpcc_log2", physname = "/dev/TClog2",
        vdevno = 48, size = 409600
```

```
use tpcc
go
checkpoint
go

if exists ( select name from sysobjects where name = 'warehouse' )
        drop table warehouse
go
create table warehouse (
        w_id        smallint,
        w_name      char(10),
        w_street_1char(20),
        w_street_2char(20),
        w_city      char(20),
        w_state     char(2),
        w_zip       char(9),
        w_tax       real,
        w_ytd       float          /*- Updated by PID, PNM */
) on Scache
go

if exists ( select name from sysobjects where name = 'district' )
        drop table district
go
create table district (
        d_id        tinyint,
        d_w_id      smallint,
        d_name      char(10),
        d_street_1char(20),
        d_street_2char(20),
        d_city      char(20),
        d_state     char(2),
        d_zip       char(9),
        d_tax       real,
        d_ytd       float,         /*- Updated by PID, PNM */
        d_next_o_idint   /*- Updated by NO */
) on Scache
go

if exists ( select name from sysobjects where name = 'customer' )
        drop table customer
go
create table customer (
        c_id        int,
        c_d_id      tinyint,
        c_w_id      smallint,
        c_first     char(16),
        c_middlechar(2),
        c_last      char(16),
        c_street_1char(20),
        c_street_2char(20),
        c_city      char(20),
        c_state     char(2),
        c_zip       char(9),
        c_phone     char(16),
        c_since     datetime,
        c_creditchar(2),
        c_credit_limnumeric(12,2),
        c_discountreal,
        c_delivery_cntsmallint,
        c_payment_cntsmallint,/*- Updated by PNM, PID */
        c_balancefloat,   /*- Updated by PNM, PID */
        c_ytd_paymentfloat,/*- Updated by PNM, PID */
        c_data1     char(250),/*- Updated (?) by PNM, PID */
        c_data2     char(250)/*- Updated (?) by PNM, PID */
) on Scustomer
go
create unique clustered index c_clu
        on customer(c_w_id, c_id, c_d_id)
        on Scustomer
go

if exists ( select name from sysobjects where name = 'history' )
        drop table history
go
create table history (
        h_c_id      int,
        h_c_d_idtinyint,
        h_c_w_idsmallint,
        h_d_id      tinyint,
        h_w_id      smallint,
        h_date      datetime,
        h_amountfloat,
        h_data      char(24)
) on Shistory
go
alter table history partition 8
go

if exists ( select name from sysobjects where name = 'new_order' )
        drop table new_order
go
create table new_order (
        no_o_id     int,
        no_d_id     tinyint,
        no_w_id     smallint,
) on Scache
go
create unique clustered index no_clu
        on new_order(no_w_id, no_d_id, no_o_id)
        on Scache
go
dbcc tune(ascinserts, 1, new_order)
go
dbcc tune(oamtrips, 100, new_order)
go

if exists ( select name from sysobjects where name = 'orders' )
        drop table orders
go
create table orders (
        o_id        int,
        o_c_id      int,
        o_d_id      tinyint,
        o_w_id      smallint,
        o_entry_ddatetime,
        o_carrier_idsmallint,/*- Updated by D */
        o_ol_cnttinyint,
        o_all_localtinyint
) on Sorders
go
create unique clustered index o_clu
        on orders(o_w_id, o_d_id, o_id)
        on Sorders
go
dbcc tune(ascinserts, 1, orders)
```

```
DEVICE  stock18         /dev/stock18  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock19         /dev/stock19  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock20         /dev/stock20  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock21         /dev/stock21  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock22         /dev/stock22  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock23         /dev/stock23  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock24         /dev/stock24  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  history01       /dev/history01    320
                db=tpcc size=320
                        segment=Shistory
DEVICE_END

**************************************************
#! /bin/sh -p

set -x
rm /dev/TC*
rm /dev/customer*
rm /dev/stock*
rm /dev/order*
rm /dev/history*

ln -s /dev/rdsk/c1t5d3s3 /dev/TCrest

ln -s /dev/rdsk/c1t0d0s3 /dev/customer01
ln -s /dev/rdsk/c1t1d0s3 /dev/customer02
ln -s /dev/rdsk/c1t2d0s3 /dev/customer03
ln -s /dev/rdsk/c1t3d0s3 /dev/customer04
ln -s /dev/rdsk/c1t4d0s3 /dev/customer05
ln -s /dev/rdsk/c1t5d0s3 /dev/customer06
ln -s /dev/rdsk/c2t0d0s3 /dev/customer07
ln -s /dev/rdsk/c2t1d0s3 /dev/customer08
ln -s /dev/rdsk/c2t2d0s3 /dev/customer09
ln -s /dev/rdsk/c2t3d0s3 /dev/customer10
ln -s /dev/rdsk/c2t4d0s3 /dev/customer_idx01

ln -s /dev/rdsk/c1t0d1s3 /dev/stock01
ln -s /dev/rdsk/c1t1d1s3 /dev/stock02
ln -s /dev/rdsk/c1t2d1s3 /dev/stock03
ln -s /dev/rdsk/c1t3d1s3 /dev/stock04
ln -s /dev/rdsk/c1t4d1s3 /dev/stock05
ln -s /dev/rdsk/c1t5d1s3 /dev/stock06
ln -s /dev/rdsk/c2t0d1s3 /dev/stock07
ln -s /dev/rdsk/c2t1d1s3 /dev/stock08
ln -s /dev/rdsk/c2t2d1s3 /dev/stock09
ln -s /dev/rdsk/c2t3d1s3 /dev/stock10
ln -s /dev/rdsk/c2t4d1s3 /dev/stock11
ln -s /dev/rdsk/c2t5d1s3 /dev/stock12
ln -s /dev/rdsk/c1t0d2s3 /dev/stock13
ln -s /dev/rdsk/c1t1d2s3 /dev/stock14
ln -s /dev/rdsk/c1t2d2s3 /dev/stock15
ln -s /dev/rdsk/c1t3d2s3 /dev/stock16
ln -s /dev/rdsk/c1t4d2s3 /dev/stock17
ln -s /dev/rdsk/c1t5d2s3 /dev/stock18
ln -s /dev/rdsk/c1t4d4s3 /dev/stock19
ln -s /dev/rdsk/c1t5d4s3 /dev/stock20
ln -s /dev/rdsk/c2t3d3s3 /dev/stock21
ln -s /dev/rdsk/c1t3d4s3 /dev/stock22
ln -s /dev/rdsk/c2t5d0s3 /dev/stock23
ln -s /dev/rdsk/c2t5d4s3 /dev/stock24


ln -s /dev/rdsk/c2t0d2s3 /dev/orderline01
ln -s /dev/rdsk/c2t1d2s3 /dev/orderline02
ln -s /dev/rdsk/c2t2d2s3 /dev/orderline03
ln -s /dev/rdsk/c2t3d2s3 /dev/orderline04
ln -s /dev/rdsk/c2t4d2s3 /dev/orderline05
ln -s /dev/rdsk/c2t5d2s3 /dev/orderline06

ln -s /dev/rdsk/c1t4d3s3 /dev/history01

ln -s /dev/rdsk/c1t0d3s3 /dev/orders01
ln -s /dev/rdsk/c1t1d3s3 /dev/orders02
ln -s /dev/rdsk/c1t2d3s3 /dev/orders03
ln -s /dev/rdsk/c2t1d4s3 /dev/orders04

ln -s /dev/rdsk/c2t0d3s3 /dev/TClog1
ln -s /dev/rdsk/c2t1d3s3 /dev/TClog2
ln -s /dev/rdsk/c2t2d3s3 /dev/TClog3

ln -s /dev/rdsk/c1t0d4s3 /dev/TCmirror1
ln -s /dev/rdsk/c1t1d4s3 /dev/TCmirror2
ln -s /dev/rdsk/c1t2d4s3 /dev/TCmirror3

ln -s /dev/rdsk/c2t4d3s3 /dev/TCtempdb

chown  -h sybase /dev/TC* /dev/customer* /dev/stock* /dev/order*
/dev/history*

**************************************************
#!/bin/sh -f

isql -Usa -P$PASSWORD << EOF
/* This script will create all the tables required for TPC-C benchmark */
/* It will also create some of the indexes. */
sp_dboption tpcc,"select into/bulkcopy",true
go
```

```
DEVICE_END

DEVICE  customer03      /dev/customer03  243
        db=tpcc size=243
                segment=Scustomer
DEVICE_END

DEVICE  customer04      /dev/customer04  243
        db=tpcc size=243
                segment=Scustomer
DEVICE_END

DEVICE  customer05      /dev/customer05  243
        db=tpcc size=243
                segment=Scustomer
DEVICE_END

DEVICE  customer06      /dev/customer06  243
        db=tpcc size=243
                segment=Scustomer
DEVICE_END

DEVICE  customer07      /dev/customer07  243
        db=tpcc size=243
                segment=Scustomer
DEVICE_END

DEVICE  customer08      /dev/customer08  243
        db=tpcc size=243
                segment=Scustomer
DEVICE_END

DEVICE  customer09      /dev/customer09  243
        db=tpcc size=243
                segment=Scustomer
DEVICE_END

DEVICE  customer10      /dev/customer10  243
        db=tpcc size=243
                segment=Scustomer
DEVICE_END

DEVICE  stock01         /dev/stock01  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock02         /dev/stock02  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock03         /dev/stock03  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock04         /dev/stock04  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock05         /dev/stock05  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock06         /dev/stock06  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock07         /dev/stock07  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock08         /dev/stock08  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock09         /dev/stock09  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock10         /dev/stock10  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock11         /dev/stock11  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock12         /dev/stock12  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock13         /dev/stock13  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock14         /dev/stock14  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock15         /dev/stock15  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock16         /dev/stock16  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END

DEVICE  stock17         /dev/stock17  170
        db=tpcc size=170
                segment=Sstock
DEVICE_END
```

Appendix B: Database Design

This Appendix contains the scripts used to create the database and the load program used to load the database initially.

```
# scale=116

DEVICE  master   /dev/TCrest  250
        db=tpcc size=250
                segment=default segment=system
                segment=Scache
DEVICE_END

DEVICE  tpcc_log        /dev/TClog1  800
        db=tpcc size=800 log
DEVICE_END

DEVICE  order_line01      /dev/orderline01  415
        db=tpcc size=415
                segment=Sorder_line
DEVICE_END

DEVICE  order_line02      /dev/orderline02  415
        db=tpcc size=415
                segment=Sorder_line
DEVICE_END

DEVICE  order_line03      /dev/orderline03  415
        db=tpcc size=415
                segment=Sorder_line
DEVICE_END

DEVICE  order_line04      /dev/orderline04  415
        db=tpcc size=415
                segment=Sorder_line
DEVICE_END

DEVICE  order_line05      /dev/orderline05  415
        db=tpcc size=415
                segment=Sorder_line
DEVICE_END

DEVICE  order_line06      /dev/orderline06  415
        db=tpcc size=415
                segment=Sorder_line
DEVICE_END

DEVICE  orders01   /dev/orders01  30
        db=tpcc size=30
                segment=Sorders
DEVICE_END

DEVICE  orders02   /dev/orders02  30
        db=tpcc size=30
                segment=Sorders
DEVICE_END

DEVICE  orders03   /dev/orders03  30
        db=tpcc size=30
                segment=Sorders
DEVICE_END

DEVICE  orders04   /dev/orders04  50
        db=tpcc size=50
                segment=Sorders
DEVICE_END

DEVICE  c_idx01 /dev/customer_idx01  190
        db=tpcc size=190
                segment=Scidx
DEVICE_END

DEVICE  customer01       /dev/customer01  243
        db=tpcc size=243
                segment=Scustomer
DEVICE_END

DEVICE  customer02       /dev/customer02  243
        db=tpcc size=243
                segment=Scustomer
```

```
 */
int
init_stock_tx()
{
    /* Install the error and message handler */
userlog("before dberrhandle \n");
        dberrhandle(err_handler);
        dbmsghandle(msg_handler);

        /* Initialize global variable for error handling */
        deadlock = 0;
        xact_type = XACT_STOCK;

userlog("before dblogin \n");
        login = dblogin();
userlog("before DBSETLUSER \n");
        DBSETLUSER(login, USER);

userlog("before DBSETLCHARSET \n");
        DBSETLCHARSET(login, getenv("CHARSET"));

    /* Open a dbproc */

userlog("before dbopen \n");
        if ((dbproc = dbopen(login, (char *)SERVER  )) == NULL)
        {
                initerr("Fatal dbopen: Could not open connection\n");
                return(-1);
        }

    /* Use the the right database */
userlog("before dbuse \n");
        if ( dbuse(dbproc, (char *)DATABASE) != SUCCEED)
        {
                initerr("Fatal dbuse: Could not use DATABASE\n");
                return(-1);
        }

    /* Done with initialization */
userlog("leaving tpsvrinit \n");
        return(0);

}


 /*
 * Function: do stocklevel transaction
 * Input is the stocklevel structure. Output is low_stock field
 */
stocklevel_tx(rqst)
TPSVCINFO *rqst;
{

        stocklevel = (struct stock_inf *)(rqst->data);

        global_w_id = stocklevel->w_id;
        global_d_id = stocklevel->d_id;
        threshold = stocklevel->threshold;

        stock_level_rpc();

        tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct stock_inf), 0);
}
```

```
/* Tuxedo */
tpsvrinit(argc, argv)
char **argv;
{
        return(init_stock_tx());/* Prepare transaction */
}

void
tpsvrdone()
{
        dbexit();
}


STOCK(rqst)
TPSVCINFO *rqst;
{
        stocklevel_tx(rqst);
}


initerr(str)
char *str;
{
        userlog("init_stock_tx ERROR during %s\n", str);
}
```

```
		byid = TRUE;
		xact_type = XACT_PAYM_ID;
	}

	if (byid) {		/* Customer selected by id */
		c_id = payp->c_id;
		payment_byid_rpc();
	}
	else {
		strcpy(c_last, payp->c_last);
		payment_byname_rpc();
		payp->c_id = c_id;
	}

	strcpy(payp->h_date, h_date);
	strcpy(payp->w_street_1, w_street_1);
	strcpy(payp->w_street_2, w_street_2);
	strcpy(payp->w_city, w_city);
	strcpy(payp->w_state, w_state);
	strcpy(payp->w_zip, w_zip);

	strcpy(payp->d_street_1, d_street_1);
	strcpy(payp->d_street_2, d_street_2);
	strcpy(payp->d_city, d_city);
	strcpy(payp->d_state, d_state);
	strcpy(payp->d_zip, d_zip);

	strcpy(payp->c_first, c_first);
	strcpy(payp->c_middle, c_middle);
	strcpy(payp->c_last, c_last);
	strcpy(payp->c_street_1, c_street_1);
	strcpy(payp->c_street_2, c_street_2);
	strcpy(payp->c_city, c_city);
	strcpy(payp->c_state, c_state);
	strcpy(payp->c_zip, c_zip);
	strcpy(payp->c_phone, c_phone);
	strcpy(payp->c_since, c_since);
	strcpy(payp->c_credit, c_credit);

	payp->c_credit_lim = c_credit_lim;
	payp->c_discount = c_discount;
	payp->c_balance = c_balance;

	if ( c_data == 0 ) {
		payp->c_data_1[0] =
		payp->c_data_2[0] =
		payp->c_data_3[0] =
		payp->c_data_4[0] = 0;
	}
	else {
		strncpy(payp->c_data_1, c_data, 50);
		strncpy(payp->c_data_2, c_data + 50, 50);
		strncpy(payp->c_data_3, c_data + 100, 50);
		strncpy(payp->c_data_4, c_data + 150, 50);
	}

	tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct pay_inf), 0);
}

/* Tuxedo code */
tpsvrinit(argc, argv)
char **argv;
{
	return(init_paym_tx());/* Prepare transaction */
}

void
tpsvrdone()
{
	dbexit();
}

PAYM(rqst)
TPSVCINFO *rqst;
{
	payment_tx(rqst);
}



initerr(str)
char *str;
{
	userlog("init_paym_tx ERROR during %s\n", str);
}

/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>
#include <time.h>
#include <signal.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

/* Sybase header files */
#include <sybfront.h>
#include <sybdb.h>
#include "SYB_tpcc.h"
#include "SYB_driver.h"



/* Tuxedo */
#include "atmi.h"
#include "userlog.h"

struct stock_inf {
int	w_id;
int d_id;
int threshold;
int low_stock;
};

struct stock_inf *stocklevel;/* Input to stocklevel transaction */

DBPROCESS		*dbproc;
LOGINREC		*login;

/*
 * Initialize transaction
```

```
#include <stdlib.h>

/* Sybase header files */
#include <sybfront.h>
#include <sybdb.h>
#include "SYB_tpcc.h"
#include "SYB_driver.h"

/* Tuxedo */
#include "atmi.h"
#include "userlog.h"

struct pay_inf {
int    w_id;
int    d_id;
int    c_id;
int    c_w_id;
int    c_d_id;
double h_amount;
double c_credit_lim;
double c_balance;
double c_discount;
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[11];
char d_street_1[21];
char d_street_2[21];
char d_city[21];
char d_state[3];
char d_zip[11];
char c_first[17];
char c_middle[3];
char c_last[17];
char c_street_1[21];
char c_street_2[21];
char c_city[21];
char c_state[3];
char c_zip[11];
char c_phone[17];
char c_since[11];
char c_credit[3];
char c_data_1[51];
char c_data_2[51];
char c_data_3[51];
char c_data_4[51];
};

struct pay_inf *payp;/* Input structure to payment_tx */

DBPROCESS       *dbproc;
LOGINREC        *login;




/*
 * Function: init payment transaction
 * Prepare the payment transaction
 */
int
init_paym_tx()
{
        /* Install the error and message handler */
userlog("before dberrhandle \n");
            dberrhandle(err_handler);
            dbmsghandle(msg_handler);

        /* Initialize global variable for error handling */
        deadlock = 0;

userlog("before dblogin \n");
            login = dblogin();
userlog("before DBSETLUSER \n");
            DBSETLUSER(login, USER);

userlog("before DBSETLPACKET \n");
            DBSETLPACKET(login, 4096);

userlog("before DBSETLCHARSET \n");
            DBSETLCHARSET(login, getenv("CHARSET"));

        /* Open a dbproc */
userlog("before dbopen \n");
            if ((dbproc = dbopen(login, (char *)SERVER  )) == NULL)
            {
                    initerr("Fatal dbopen: Could not open connection\n");
                    return(-1);
            }

        /* Use the the right database */
userlog("before dbuse \n");
            if ( dbuse(dbproc, (char *)DATABASE) != SUCCEED)
            {
                    initerr("Fatal dbuse: Could not use DATABASE\n");

                    return(-1);
            }

        /* Done with initialization */
userlog("leaving tpsvrinit \n");
        return(0);

}


payment_tx(rqst)
TPSVCINFO *rqst;
{
        int     byid;

        payp = (struct pay_inf *)(rqst->data);


        global_w_id = payp->w_id;
        c_w_id = payp->c_w_id;
        h_amount = payp->h_amount;
        global_d_id = payp->d_id;
        c_d_id = payp->c_d_id;
        if (payp->c_id == 0) {             /* Customer selected by name */
                byid = FALSE;
                xact_type = XACT_PAYM_NAME;
        }
        else {
```

```
struct ord_inf *ordstat;/* Input structure to ordstat_tx */

DBPROCESS       *dbproc;
LOGINREC        *login;

/*
 * Function: init ordstat transaction
 * Prepare the ordstat transaction
 */
int
init_ords_tx()
{
        /* Install the error and message handler */
userlog("before dberrhandle \n");
            dberrhandle(err_handler);
            dbmsghandle(msg_handler);

        /* Initialize global variable for error handling */
        deadlock = 0;

userlog("before dblogin \n");
            login = dblogin();
userlog("before DBSETLUSER \n");
            DBSETLUSER(login, USER);

userlog("before DBSETLPACKET \n");
            DBSETLPACKET(login, 4096);

userlog("before DBSETLCHARSET \n");
            DBSETLCHARSET(login, getenv("CHARSET"));

        /* Open a dbproc */
userlog("before dbopen \n");
            if ((dbproc = dbopen(login, (char *)SERVER  )) == NULL)
            {
                    initerr("Fatal dbopen: Could not open connection\n");
                    return(-1);
            }

        /* Use the the right database */
userlog("before dbuse \n");
            if ( dbuse(dbproc, (char *)DATABASE) != SUCCEED)
            {
                    initerr("Fatal dbuse: Could not use DATABASE\n");
                    return(-1);
            }

        /* Done with initialization */
userlog("leaving tpsvrinit \n");
            return(0);


}


ordstat_tx(rqst)
TPSVCINFO *rqst;
{
        int     byid;

        ordstat = (struct ord_inf *)(rqst->data);
        if (ordstat->c_id == 0) {       /* Customer selected by name */
                byid = FALSE;
                xact_type = XACT_ORDS_NAME;
        }
        else {
                byid = TRUE;
                xact_type = XACT_ORDS_ID;
        }

        c_w_id = ordstat->w_id;
        c_d_id = ordstat->d_id;
        if (!byid) {
                strcpy(c_last, ordstat->c_last);
                order_status_byname_rpc();
                ordstat->c_id = c_id;
        }
        else {
                c_id = ordstat->c_id;
                order_status_byid_rpc();
                strcpy(ordstat->c_last, c_last);
        }


                tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct ord_inf), 0);
}


tpsvrinit(argc, argv)
char **argv;
{
        return(init_ords_tx());/* Prepare transaction */
}

void
tpsvrdone()
{
        dbexit();
}

ORDS(rqst)
TPSVCINFO *rqst;
{
        ordstat_tx(rqst);
}


initerr(str)
char *str;
{
        userlog("init_ordstat_tx ERROR during %s\n", str);
}


/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>
#include <time.h>
#include <signal.h>
#include <math.h>
#include <string.h>
```

```
userlog("before DBSETLPACKET \n");
        DBSETLPACKET(login, 4096);

userlog("before DBSETLCHARSET \n");
        DBSETLCHARSET(login, getenv("CHARSET"));

        /* Open a dbproc */
userlog("before dbopen \n");
        if ((dbproc = dbopen(login, (char *)SERVER  )) == NULL)
        {
                initerr("Fatal dbopen: Could not open connection\n");
                return(-1);
        }

        /* Use the the right database */
userlog("before dbuse \n");
        if ( dbuse(dbproc, (char *)DATABASE) != SUCCEED)
        {
                initerr("Fatal dbuse: Could not use DATABASE\n");
                return(-1);
        }

        /* Done with initialization */
userlog("leaving tpsvrinit \n");
        return(0);

}


/*
 * This function executes the neworder transaction
 */
neworder_tx(rqst)
TPSVCINFO *rqst;
{

        int     i;
int rollback = 0;
int linecnt;
        int ret;
struct items_inf *cur_ip;       /* Pointer to current item */


        neworder = (struct newo_inf *)(rqst->data);
        linecnt = neworder->o_ol_cnt;

again:
        neworder->total = 0;

        global_w_id = neworder->w_id;
        global_d_id = neworder->d_id;
        c_id = neworder->c_id;
        o_ol_cnt = neworder->o_ol_cnt;
        o_all_local = 1;

        for (i = 0; i < (int)o_ol_cnt ; i++) {
                cur_ip = &neworder->n_items[i];

                ol[i].i_id = cur_ip->ol_i_id;
                ol[i].supply_w_id = cur_ip->ol_supply_w_id;
                ol[i].quantity = cur_ip->ol_quantity;
                if (ol[i].supply_w_id != global_w_id)
                        o_all_local = 0;        /* non-local order */
        }

        new_order_rpc();


        neworder->total = total_amount;

        tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct newo_inf), 0);
}

/* Start of Tuxedo code */
int
tpsvrinit(argc, argv)
char **argv;
{
        return(init_newo_tx());/* Prepare transaction */
}

void
tpsvrdone()
{
        dbexit();
}

NEWO(rqst)
TPSVCINFO *rqst;
{
        neworder_tx(rqst);
}


initerr(str)
char *str;
{
        userlog("SQL ERROR during %s\n", str);
}

/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */


#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>
#include <time.h>
#include <signal.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

/* Sybase header files */
#include <sybfront.h>
#include <sybdb.h>
#include "SYB_tpcc.h"
#include "SYB_driver.h"

/* Tuxedo */
#include "atmi.h"
#include "userlog.h"
#include "tpcc_tux_forms.h"
```

```
delivery_tx(rqst)
TPSVCINFO *rqst;
{

        delp = (struct req_struct *)(rqst->data);
        global_w_id = delp->w_id;
        o_carrier_id = delp->o_carrier_id;
        tx_count++;
        sprintf(outbuf, "Starting transaction %d queued at %d\n",
                tx_count, delp->qtime);
        delivery_rpc();     /* XXX: use Sybase's SYB_rpc.c version  */


        sprintf(outbuf+strlen(outbuf), "Transaction completed at %d\n",
time(0));
        fwrite(outbuf, strlen(outbuf), 1, delfile);
        fflush(delfile);
        tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct req_struct), 0);
}


/* If errors occur during initialization, exit */
initerr(str)
char *str;
{
        userlog("init_delivery_tx ERROR %\n", str);
}


/* Tuxedo */
tpsvrinit(argc, argv)
char **argv;
{
        char *p, ident[20];
        char filename[200];
        int proc_no, count;
        struct utsname name;


        if ((p = getenv("TMPDIR")) == (char *)NULL) {
                userlog("TMPDIR environment variable not set\n");
                exit(1);
        }

        proc_no = atoi(argv[optind]);/* Needs argument which is the
proc_no */

        /* Get hostname of our machine  and create results file */
        uname( &name);
        strcpy(filename, p);
        sprintf(filename+strlen(filename), "/%s.del%d", name.nodename,
proc_no);
        userlog("filename = %s \n",filename);
        delfile = fopen(filename, "w");
        if (delfile == NULL) {
                userlog("Cannot create file %s\n", filename);
        }
        return(init_del_tx());/* Prepare transaction */
}

void
tpsvrdone()
{
        cleanup();      /* Close results file */
        dbexit();
}

DEL(rqst)
TPSVCINFO *rqst;
{
        delivery_tx(rqst);
}

/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */


#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>
#include <time.h>
#include <signal.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

/* Sybase header files */
#include <sybfront.h>
#include <sybdb.h>
#include "SYB_tpcc.h"
#include "SYB_driver.h"

/* Tuxedo includes */
#include "atmi.h"
#include "userlog.h"

#include "tpcc_tux_forms.h"


struct newo_inf *neworder;/* Neworder field structure */
char blank_mesg[25] = "                         ";

DBPROCESS       *dbproc;
LOGINREC        *login;


/*
 * Initialize the neworder transaction
 */
int
init_newo_tx()
{
        /* Install the error and message handler */
userlog("before dberrhandle \n");
        dberrhandle(err_handler);
        dbmsghandle(msg_handler);

        /* Initialize global variable for error handling */
        deadlock = 0;
        xact_type = XACT_NEWO;

userlog("before dblogin \n");
        login = dblogin();
userlog("before DBSETLUSER \n");
        DBSETLUSER(login, USER);
```

```
                select     /* Return to client */
                        @no_o_id,
                        convert(char(11), @ol_delivery_d, 105)
                        + convert(char(8), @ol_delivery_d, 108)
        end
end
go
if exists ( SELECT name FROM sysobjects WHERE name = 'stock_level')
    DROP PROC stock_level
go

CREATE PROC stock_level
        @w_idsmallint,
        @d_idtinyint,
        @threshold smallint
as
        select  count(distinct(s_i_id))  /* Return to client */
        FROMdistrict,
                order_line,
                stock
        WHEREd_w_id=@w_id
        AND d_id  =      @d_id
        AND ol_w_id=   @w_id
        AND ol_d_id=   @d_id
        AND ol_o_idbetween (d_next_o_id - 20) and (d_next_o_id - 1)
        AND s_w_id=     ol_w_id
        AND s_i_id=      ol_i_id
        AND      s_quantity < @threshold
go
EOF

/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */


#include "tpcc_client.h"
#include <stdlib.h>
#include <sys/signal.h>
#include <sys/utsname.h>
#include <errno.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>
#include <time.h>


/* Tuxedo */
#include "atmi.h"
#include "userlog.h"

/* Sybase header files */
#include <sybfront.h>
#include <sybdb.h>
#include "SYB_tpcc.h"
#include "SYB_driver.h"


static struct req_struct *delp;/* Transaction message */
extern char outbuf[];
extern int tx_count;/* Transaction counter */
extern FILE *delfile;

DBPROCESS      *dbproc;
LOGINREC       *login;


cleanup()
{
        fclose(delfile);
}


int
init_del_tx()
{
        /* Prepare delivery transaction */

        /* Install the error and message handler */
    userlog("before dberrhandle \n");
        dberrhandle(err_handler);
        dbmsghandle(msg_handler);

        /* initialize global variable for deadlock and error handling */
        deadlock = 0;
        xact_type = XACT_BKEND;

    userlog("before dblogin \n");
        login = dblogin();
    userlog("before DBSETLUSER \n");
        DBSETLUSER(login, USER);

    userlog("before DBSETLPACKET \n");

        DBSETLPACKET(login, 4096);


    userlog("before DBSETLCHARSET \n");
        DBSETLCHARSET(login, getenv("CHARSET"));

        /* Open a dbproc */
    userlog("before dbopen \n");
        if ((dbproc = dbopen(login, (char *)SERVER  )) == NULL)
        {
            initerr("Fatal dbopen: Could not open connection\n");
            return(-1);
        }

        /* Use the the right database */
    userlog("before dbuse \n");
        if ( dbuse(dbproc, (char *)DATABASE) != SUCCEED)
        {
            initerr("Fatal dbuse: Could not use DATABASE\n");
            return(-1);
        }

        /* Done with initialization */
    userlog("leaving tpsvrinit \n");
        return(0);

}
```

```
        FROM customer (index c_non1) HOLDLOCK
        WHEREc_w_id = @w_id and
              c_d_id = @d_id and
              c_last = @c_last
        OPEN c_find
        while (@n>0) begin
              FETCH c_find INTO @c_id
              SELECT @n = @n-1
        end
        CLOSE c_find

  /* Get the latest order made by the customer */
        SELECT@o_id = o_id, @o_carrier_id = o_carrier_id,
              @o_entry_d = o_entry_d
        FROMorders (index o_clu prefetch 16 lru) HOLDLOCK
        WHEREo_w_id= @w_id
        AND o_d_id= @d_id
        AND o_c_id= @c_id

  /* Select order lines for the current order */
        select /* Return multiple rows to client */
              ol_supply_w_id,
              ol_i_id,
              ol_quantity,
              ol_amount,
              convert(char(10), ol_delivery_d, 105)
        FROMorder_line HOLDLOCK
        WHEREol_o_id = @o_id
        AND ol_d_id = @d_id
        AND ol_w_id = @w_id

        select /* Return single row to client */
              @c_id, c_last, c_first, c_middle, c_balance,
              @o_id,
              convert(char(11),@o_entry_d,105)
              + convert(char(8),@o_entry_d,108),
              @o_carrier_id
        FROMcustomer HOLDLOCK
        WHERE   c_id    = @c_id
        AND c_d_id  = @d_id
        AND c_w_id  = @w_id

  COMMIT TRANSACTION OSNM
  go
  if exists (select * from sysobjects where name = 'delivery')
        drop proc delivery
  go
  CREATE PROC delivery
        @w_id       smallint,
        @d_id       tinyint,
        @o_carrier_idsmallint
  as

  declare@no_o_id int,  @o_c_id     smallint,
        @ol_total float,   @ol_amount float,
        @ol_delivery_d datetime,@junk_idsmallint,
        @today       datetime,@1       smallint

  declare c_del_no CURSOR FOR
        SELECTno_o_id
        FROMnew_order (index no_clu) HOLDLOCK
        WHEREno_d_id = @d_id
        AND no_w_id = @w_id
        FOR UPDATE

  declare c_del_ol CURSOR FOR
        SELECTol_amount, ol_delivery_d
        FROMorder_line HOLDLOCK
        WHEREol_o_id= @no_o_id
        AND ol_d_id= @d_id
        AND ol_w_id= @w_id
        FOR UPDATE OF ol_delivery_d

  declare c_del_o CURSOR FOR
        SELECTo_c_id, o_carrier_id
        FROMorders HOLDLOCK
        WHEREo_id= @no_o_id
        AND o_d_id= @d_id
        AND o_w_id= @w_id
        FOR UPDATE OF o_carrier_id

  begin
        select @1 = 1
        BEGIN TRANSACTION DEL
        OPEN c_del_no
        FETCH c_del_no INTO @no_o_id

        if (@@sqlstatus != 0)
        begin
              COMMIT TRANSACTION DEL
              select NULL, NULL
        end
        else
        begin
              DELETEFROM new_order
              WHERECURRENT OF c_del_no
              CLOSEc_del_no

              SELECT@ol_total = 0.0, @today = getdate()
              OPENc_del_ol
              FETCHc_del_ol INTO @ol_amount, @ol_delivery_d
              while (@@sqlstatus = 0)
              begin
                    SELECT@ol_total = @ol_total + @ol_amount
                    UPDATEorder_line
                    SET   ol_delivery_d = @today
                    WHERECURRENT OF c_del_ol
                    FETCHc_del_ol INTO @ol_amount, @ol_delivery_d
              end
              CLOSE c_del_ol

              OPENc_del_o
              FETCHc_del_o INTO @o_c_id, @junk_id
              UPDATEorders
              SET   o_carrier_id = @o_carrier_id
              WHERECURRENT OF c_del_o
              CLOSEc_del_o

              UPDATEcustomer
              SET     c_balance= c_balance + @ol_total,
                    c_delivery_cnt= c_delivery_cnt + @1
              WHERE   c_id= @o_c_id
              AND     c_d_id= @d_id
              AND     c_w_id= @w_id

              COMMIT TRANSACTION DEL
        end
        else begin
                SELECT@screen_data = NULL
                UPDATEcustomer SET
                        c_payment_cnt= @c_payment_cnt,
                        c_balance= @c_balance,
                        c_ytd_payment= @c_ytd_payment
                        WHERE CURRENT OF c_pay_c
        end
        CLOSE c_pay_c

        /* Create the history record */
        SELECT @today = getdate()
        INSERT INTO history (
                h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
                h_date, h_amount, h_data)
        VALUES (
                @c_id, @c_d_id, @c_w_id, @c_d_id, @w_id,
                @today, @h_amount, (@w_name + "     " + @d_name))

COMMIT TRANSACTION PNM

        select          /* Return to client */
                @c_id,
                @c_last,
                convert(char(11),@today,105) + convert(char(8),@today,108),
                @w_street_1,
                @w_street_2,
                @w_city,
                @w_state,
                @w_zip,

                @d_street_1,
                @d_street_2,
                @d_city,
                @d_state,
                @d_zip,

                @c_first,
                @c_middle,
                @c_street_1,
                @c_street_2,
                @c_city,
                @c_state,
                @c_zip,
                @c_phone,
                convert(char(10), @c_since, 105),
                @c_credit,
                @c_credit_lim,
                @c_discount,
                @c_balance,
                @screen_data
go
if exists (select * from sysobjects where name = 'order_status_byid')
        DROP PROC order_status_byid
go
CREATE PROC order_status_byid
        @w_id           smallint,
        @d_id           tinyint,
        @c_id           int
as

DECLARE@o_id int,
        @o_entry_ddatetime,
        @o_carrier_idsmallint

BEGIN TRANSACTION OSID

/* Get the latest order made by the customer */
        SELECT@o_id = o_id, @o_carrier_id = o_carrier_id,
                @o_entry_d = o_entry_d
        FROMorders (index o_clu prefetch 16 lru) HOLDLOCK
        WHEREo_w_id= @w_id
        AND o_d_id= @d_id
        AND o_c_id= @c_id

/* Select order lines for the current order */
        select /* Return multiple rows to client */
                ol_supply_w_id,
                ol_i_id,
                ol_quantity,
                ol_amount,
                convert(char(10), ol_delivery_d, 105)
        FROMorder_line HOLDLOCK
        WHEREol_o_id = @o_id
        AND ol_d_id = @d_id
        AND ol_w_id = @w_id

        select /* Return single row to client */
                @c_id, c_last, c_first, c_middle, c_balance,
                @o_id,
                convert(char(11),@o_entry_d,105)
                + convert(char(8),@o_entry_d,108),
                @o_carrier_id
        FROMcustomer HOLDLOCK
        WHERE   c_id    = @c_id
        AND c_d_id  = @d_id
        AND c_w_id  = @w_id

COMMIT TRANSACTION OSID
go
if exists (select * from sysobjects where name = 'order_status_byname')
        DROP PROC order_status_byname
go
CREATE PROC order_status_byname
        @w_id           smallint,
        @d_id           tinyint,
        @c_lastchar(16)
as

DECLARE@o_id int,
        @o_entry_ddatetime,
        @o_carrier_idsmallint

declare@n int,    @c_idint
declare c_find CURSOR FOR
        SELECT  c_id
        FROM customer (index c_non1) HOLDLOCK
        WHERE   c_w_id  =       @w_id
        AND     c_d_id  =       @d_id
        AND     c_last  =       @c_last
        ORDER BY c_w_id, c_d_id, c_last, c_first, c_id
        FOR READ ONLY

BEGIN TRANSACTION OSNM
        SELECT @n = (count(*)+1)/2
```

```
if exists (select * from sysobjects where name = 'payment_byname')
        DROP PROC payment_byname
go
CREATE PROC payment_byname
        @w_id        smallint,@c_w_idsmallint,
        @h_amount float,
        @d_id        tinyint,@c_d_id   tinyint,
        @c_lastchar(16)
as
declare@n int,    @c_idint

declare@w_street_1char(20),@w_street_2char(20),
        @w_city      char(20),@w_statechar(2),
        @w_zip       char(9),@w_namechar(10),
        @w_ytd       float

declare@d_street_1char(20),@d_street_2char(20),
        @d_city      char(20),@d_statechar(2),
        @d_zip       char(9),@d_namechar(10),
        @d_ytd       float

declare@c_firstchar(16),@c_middlechar(2),
        @c_street_1char(20),@c_street_2char(20),
        @c_city      char(20),@c_statechar(2),
        @c_zip       char(9),@c_phonechar(16),
        @c_sincedatetime,@c_creditchar(2),
        @c_credit_limnumeric(12,2),@c_balancefloat,
        @c_discountreal, @c_ytd_paymentfloat,
        @c_payment_cntsmallint,@1smallint,
        @data1       char(250),@data2char(250),
        @c_data_1char(250),@c_data_2char(250)

declare @screen_datachar(200),@today datetime

declare c_pay_wd CURSOR FOR
        SELECTw_street_1, w_street_2, w_city,
                w_state, w_zip, w_name, w_ytd,
                d_street_1, d_street_2, d_city,
                d_state, d_zip, d_name, d_ytd
        FROMwarehouse HOLDLOCK,
                district HOLDLOCK
        WHEREd_w_id= @w_id
        AND d_id   = @d_id
        AND w_id  = d_w_id
        FOR UPDATE OF w_ytd, d_ytd

declare c_pay_c CURSOR FOR
        SELECTc_first, c_middle, c_last, c_street_1, c_street_2,
                c_city, c_state, c_zip, c_phone, c_credit, c_credit_lim,
                c_discount, c_balance, c_ytd_payment, c_payment_cnt,
                c_since, c_data1, c_data2
        FROMcustomer HOLDLOCK
        WHEREc_w_id= @c_w_id
        AND c_d_id= @c_d_id
        AND c_id   = @c_id
        FOR UPDATE OF c_balance, c_payment_cnt, c_ytd_payment,
c_data1, c_data2

declare c_find CURSOR FOR
        SELECT  c_id
        FROM customer (index c_non1) HOLDLOCK
        WHERE   c_w_id  =       @c_w_id
        AND     c_d_id  =       @c_d_id
        AND     c_last  =       @c_last
        ORDER BY c_w_id, c_d_id, c_last, c_first, c_id
        FOR READ ONLY

BEGIN TRANSACTION PNM
        SELECT @n = (count(*)+1)/2
        FROM customer (index c_non1) HOLDLOCK
        WHEREc_w_id = @c_w_id and
                c_d_id = @c_d_id and
                c_last = @c_last
        OPEN c_find
        while (@n>0) begin
                FETCH c_find INTO @c_id
                SELECT @n = @n-1
        end
        CLOSE c_find
        select @1 = 1
        OPEN c_pay_wd
        FETCH c_pay_wd INTO
                @w_street_1, @w_street_2, @w_city,
                @w_state, @w_zip, @w_name, @w_ytd,
                @d_street_1, @d_street_2, @d_city,
                @d_state, @d_zip, @d_name, @d_ytd
        UPDATE district
                SET     d_ytd   = @d_ytd + @h_amount
                WHERE CURRENT OF c_pay_wd
        UPDATE warehouse
                SET w_ytd = @w_ytd + @h_amount
                WHERE CURRENT OF c_pay_wd
        CLOSE c_pay_wd

        OPEN c_pay_c
        FETCH c_pay_c INTO
                @c_first, @c_middle, @c_last, @c_street_1, @c_street_2,
                @c_city, @c_state, @c_zip, @c_phone, @c_credit, @c_credit_lim,
                @c_discount, @c_balance, @c_ytd_payment, @c_payment_cnt,
                @c_since, @data1, @data2

        SELECT@c_payment_cnt = @c_payment_cnt + @1,
                @c_balance = @c_balance - @h_amount,
                @c_ytd_payment = @c_ytd_payment + @h_amount

        if (@c_credit = "BC")
        begin
                SELECT  @c_data_2 =
                        substring(@data1, 209, 42) +
                        substring(@data2, 1, 208)
                SELECT  @c_data_1 =
                        convert(char(5), @c_id) +
                        convert(char(4), @c_d_id) +
                        convert(char(5), @c_w_id) +
                        convert(char(4), @d_id) +
                        convert(char(5), @w_id) +
                        convert(char(19),convert(numeric(18,2), @h_amount)) +
                        substring(@data1, 1, 208)
                SELECT@screen_data = substring(@c_data_1, 1, 200)

                UPDATE customer SET
                        c_payment_cnt = @c_payment_cnt,
                        c_ytd_payment = @c_ytd_payment,
                        c_balance = @c_balance,
                        c_data1 = @c_data_1, c_data2 = @c_data_2
                WHERE CURRENT OF c_pay_c
```

```
declare c_pay_wd CURSOR FOR
        SELECTw_street_1, w_street_2, w_city,
                w_state, w_zip, w_name, w_ytd,
                d_street_1, d_street_2, d_city,
                d_state, d_zip, d_name, d_ytd
        FROMwarehouse HOLDLOCK,
                district HOLDLOCK
        WHEREd_w_id= @w_id
        AND d_id   = @d_id
        AND w_id  = d_w_id
        FOR UPDATE OF w_ytd, d_ytd

declare c_pay_c CURSOR FOR
        SELECTc_first, c_middle, c_last, c_street_1, c_street_2,
                c_city, c_state, c_zip, c_phone, c_credit, c_credit_lim,
                c_discount, c_balance, c_ytd_payment, c_payment_cnt,
                c_since, c_data1, c_data2
        FROMcustomer HOLDLOCK
        WHEREc_w_id= @c_w_id
        AND c_d_id= @c_d_id
        AND c_id   = @c_id
        FOR UPDATE OF c_balance, c_payment_cnt, c_ytd_payment,
c_data1, c_data2

BEGIN TRANSACTION PID
        select @1 = 1
        OPEN c_pay_wd
        FETCH c_pay_wd INTO
                @w_street_1, @w_street_2, @w_city,
                @w_state, @w_zip, @w_name, @w_ytd,
                @d_street_1, @d_street_2, @d_city,
                @d_state, @d_zip, @d_name, @d_ytd
        UPDATE district
                SET     d_ytd   = @d_ytd + @h_amount
                WHERE CURRENT OF c_pay_wd
        UPDATE warehouse
                SET w_ytd = @w_ytd + @h_amount
                WHERE CURRENT OF c_pay_wd
        CLOSE c_pay_wd

        OPEN c_pay_c
        FETCH c_pay_c INTO
                @c_first, @c_middle, @c_last, @c_street_1, @c_street_2,
                @c_city, @c_state, @c_zip, @c_phone, @c_credit, @c_credit_lim,
                @c_discount, @c_balance, @c_ytd_payment, @c_payment_cnt,
                @c_since, @data1, @data2

        SELECT@c_payment_cnt = @c_payment_cnt + @1,
                @c_balance = @c_balance - @h_amount,
                @c_ytd_payment = @c_ytd_payment + @h_amount

        if (@c_credit = "BC")
        begin
                SELECT  @c_data_2 =
                        substring(@data1, 209, 42) +
                        substring(@data2, 1, 208)
                SELECT  @c_data_1 =
                        convert(char(5), @c_id) +
                        convert(char(4), @c_d_id) +
                        convert(char(5), @c_w_id) +
                        convert(char(4), @d_id) +
                        convert(char(5), @w_id) +
                        convert(char(19),convert(numeric(18,2), @h_amount)) +
                        substring(@data1, 1, 208)
                SELECT@screen_data = substring(@c_data_1, 1, 200)

                UPDATE customer SET
                        c_payment_cnt = @c_payment_cnt,
                        c_ytd_payment = @c_ytd_payment,
                        c_balance = @c_balance,
                        c_data1 = @c_data_1, c_data2 = @c_data_2
                WHERE CURRENT OF c_pay_c
        end
        else begin
                SELECT@screen_data = NULL
                UPDATEcustomer SET
                        c_payment_cnt= @c_payment_cnt,
                        c_balance= @c_balance,
                        c_ytd_payment= @c_ytd_payment
                        WHERE CURRENT OF c_pay_c
        end
        CLOSE c_pay_c

        /* Create the history record */
        SELECT @today = getdate()
        INSERT INTO history (
                h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
                h_date, h_amount, h_data)
        VALUES (
                @c_id, @c_d_id, @c_w_id, @c_d_id, @w_id,
                @today, @h_amount, (@w_name + "     " + @d_name))

COMMIT TRANSACTION PID

        select          /* Return to client */
                @c_id,
                @c_last,
                convert(char(11),@today,105) + convert(char(8),@today,108),
                @w_street_1,
                @w_street_2,
                @w_city,
                @w_state,
                @w_zip,

                @d_street_1,
                @d_street_2,
                @d_city,
                @d_state,
                @d_zip,

                @c_first,
                @c_middle,
                @c_street_1,
                @c_street_2,
                @c_city,
                @c_state,
                @c_zip,
                @c_phone,
                convert(char(10), @c_since, 105),
                @c_credit,
                @c_credit_lim,
                @c_discount,
                @c_balance,
                @screen_data
go
```

```
        if (@@sqlstatus != 0) begin/* item not found */
                SELECT @commit_flag = 0
                select          /* Return to client */
                        NULL, NULL, NULL, NULL, NULL
                        break
        end

        if @d_id < @5
          if @d_id < @3
                if @d_id < @2SELECT @s_dist = @s_dist_01
                else        SELECT @s_dist = @s_dist_02
          else if @d_id < @4SELECT @s_dist = @s_dist_03
                else        SELECT @s_dist = @s_dist_04
        else if @d_id < @7
                if @d_id < @6SELECT @s_dist = @s_dist_05
                else        SELECT @s_dist = @s_dist_06
          else if @d_id < @9
                  if @d_id < @8SELECT @s_dist = @s_dist_07
                  else      SELECT @s_dist = @s_dist_08
        else if @d_id < @10SELECT @s_dist = @s_dist_09
          else              SELECT @s_dist = @s_dist_10

        select @ol_qty_smallint = @ol_qty
        if @s_quantity >= @ol_qty_smallint + @ten
                SELECT @s_quantity = @s_quantity -
@ol_qty_smallint
        else
                SELECT @s_quantity = @s_quantity -
@ol_qty_smallint + 91

        if (@s_w_id = @w_id)
                SELECT@remote = 0
        else
                SELECT@remote = 1

        UPDATE stock set
                s_quantity= @s_quantity,
                s_ytd      = @s_ytd + @ol_qty,
                s_remote_cnt= @s_remote_cnt + @remote,
                s_order_cnt= @s_order_cnt + @one
                WHERE CURRENT OF c_no_is

        if (patindex("%ORIGINAL%", @i_data) > 0) and
           (patindex("%ORIGINAL%", @s_data) > 0)
                SELECT @b_g = "B"
        else
                SELECT @b_g = "G"

    SELECT @ol_amount = @ol_qty * @i_price
        INSERT INTO order_line (
                ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
                ol_supply_w_id, ol_delivery_d, ol_quantity,
                ol_amount, ol_dist_info)
        VALUES (
                @o_id, @d_id, @w_id, @ol_number+@o_ol_done,
@i_id,
                @s_w_id, "19000101", @ol_qty_smallint,
                @ol_amount, @s_dist)

        select                  /* Return to client */
                @i_name,
                @i_price,
                @s_quantity,
                        @ol_amount,
                        @b_g
                CLOSE c_no_is
        end

        if (@o_ol_done + @o_ol_now >= @o_ol_cnt)/* Finish order */
        begin
                SELECT @o_entry_d = getdate()
                INSERT INTO orders (
                        o_id, o_c_id, o_d_id, o_w_id,
                        o_entry_d, o_carrier_id, o_ol_cnt, o_all_local)
                VALUES (
                        @o_id, @c_id, @d_id, @w_id,
                        @o_entry_d, -1, @o_ol_cnt, 0)
                INSERT INTO new_order (no_o_id, no_d_id, no_w_id)
                VALUES (@o_id, @d_id, @w_id)

                if (@o_ol_now >= @o_ol_cnt)
                        if (@commit_flag = @1)
                                commit transaction NO
                        else
                                rollback transaction NO
        end

        select          /* Return to client */
                @w_tax, @d_tax, @o_id, @c_last,
                @c_discount, @c_credit,
                convert(char(11),@o_entry_d,105) + convert(char(8),@o_entry_d,108)
end
go
if exists (select * from sysobjects where name = 'payment_byid')
        DROP PROC payment_byid
go
CREATE PROC payment_byid
        @w_id       smallint,@c_w_idsmallint,
        @h_amount float,
        @d_id       tinyint,@c_d_id  tinyint,
        @c_idint
as
declare@c_last    char(16)

declare@w_street_1char(20),@w_street_2char(20),
        @w_city     char(20),@w_statechar(2),
        @w_zip      char(9),@w_namechar(10),
        @w_ytd      float

declare@d_street_1char(20),@d_street_2char(20),
        @d_city     char(20),@d_statechar(2),
        @d_zip      char(9),@d_namechar(10),
        @d_ytd      float

declare@c_firstchar(16),@c_middlechar(2),
        @c_street_1char(20),@c_street_2char(20),
        @c_city     char(20),@c_statechar(2),
        @c_zip      char(9),@c_phonechar(16),
        @c_sincedatetime,@c_creditchar(2),
        @c_credit_limnumeric(12,2),@c_balancefloat,
        @c_discountreal, @c_ytd_paymentfloat,
        @c_payment_cntsmallint,@1smallint,
        @data1      char(250),@data2char(250),
        @c_data_1char(250),@c_data_2char(250)

declare @screen_datachar(200),@today datetime
```

```
declare
        @w_tax       real,           @d_tax       real,
        @c_last      char(16),@c_creditchar(2),
        @c_discountreal,

        @ol_amountreal,
        @i_pricereal,
        @i_name      char(24),@i_data char(50),

        @s_quantitysmallint,
        @s_ytd       int,            @s_order_cntint,
        @s_dist      char(24),@s_data char(50),
        @s_dist_01char(24),@s_dist_02char(24),
        @s_dist_03char(24),@s_dist_04char(24),
        @s_dist_05char(24),@s_dist_06char(24),
        @s_dist_07char(24),@s_dist_08char(24),
        @s_dist_09char(24),@s_dist_10char(24),
        @s_remote_cntint,@remote   int,

        @ol_numbertinyint,
        @o_entry_ddatetime,@b_gchar(1)

declare@0  tinyint,@1  tinyint,@2  tinyint,@3  tinyint,
        @4  tinyint,@5  tinyint,@6  tinyint,@7  tinyint,
        @8  tinyint,@9  tinyint,@10 tinyint,@11 tinyint,
        @12 tinyint,@13 tinyint,@14 tinyint,@15 tinyint,
        @one smallint,@ten smallint,@ol_qty_smallint smallint

declare c_no_wdc CURSOR FOR
        SELECTw_tax, d_tax, d_next_o_id,
                c_last, c_discount, c_credit
                FROMwarehouse HOLDLOCK,
                        district HOLDLOCK,
                        customer (index c_clu prefetch 2 lru) HOLDLOCK
                WHEREd_w_id= @w_id
                AND d_id  = @d_id
                AND w_id = d_w_id
                AND c_w_id= w_id
                AND c_d_id= d_id
                AND c_id  = @c_id
                FOR UPDATE OF d_next_o_id

declare c_no_is CURSOR FOR
        SELECTi_price, i_name, i_data,
                s_quantity, s_data,
                s_ytd, s_order_cnt, s_remote_cnt,/* for update */
                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
                FROMitem HOLDLOCK,
                        stock HOLDLOCK
                WHEREs_w_id = @s_w_id
                AND s_i_id = i_id
                AND   i_id = @i_id
                FOR UPDATE OF s_quantity, s_ytd, s_order_cnt, s_remote_cnt

begin
        select@0=0, @1=1, @2=2, @3=3, @4=4, @5=5, @6=6, @7=7, @8=8,
                @9=9, @10=10, @11=11, @12=12, @13=13, @14=14, @15=15,
                @one=1, @ten=10

        if (@o_ol_done = @0)/* Start a new order */
        begin
                if (@o_ol_now >= @o_ol_cnt)/* One call only */
                        begin transaction NO

                OPEN c_no_wdc
                FETCH c_no_wdc INTO
                        @w_tax, @d_tax, @o_id,
                        @c_last, @c_discount, @c_credit
                SELECT@commit_flag= 1
                UPDATEdistrict
                        SET   d_next_o_id = @o_id + 1
                        WHERE CURRENT OF c_no_wdc
                CLOSE c_no_wdc
        end

        SELECT @ol_number = @0
        while (@ol_number < @o_ol_cnt) begin
                SELECT @ol_number = @ol_number + @1

                if @ol_number < @5
                        if @ol_number < @3 begin
                                if @ol_number >= @2SELECT @i_id = @i_id2,
@s_w_id = @s_w_id2,  @ol_qty = @ol_qty2
                        end
                        else   if @ol_number < @4SELECT @i_id = @i_id3,  @s_w_id
= @s_w_id3,  @ol_qty = @ol_qty3
                                else              SELECT @i_id = @i_id4,  @s_w_id =
@s_w_id4,  @ol_qty = @ol_qty4
                else if @ol_number < @9
                        if @ol_number < @7
                                if @ol_number < @6SELECT @i_id = @i_id5,  @s_w_id
= @s_w_id5,  @ol_qty = @ol_qty5
                                else              SELECT @i_id = @i_id6,  @s_w_id =
@s_w_id6,  @ol_qty = @ol_qty6
                        else   if @ol_number < @8SELECT @i_id = @i_id7,  @s_w_id
= @s_w_id7,  @ol_qty = @ol_qty7
                                else              SELECT @i_id = @i_id8,  @s_w_id =
@s_w_id8,  @ol_qty = @ol_qty8
                else if @ol_number < @13
                        if @ol_number < @11
                                if @ol_number < @10SELECT @i_id = @i_id9,
@s_w_id = @s_w_id9,  @ol_qty = @ol_qty9
                                else              SELECT @i_id = @i_id10, @s_w_id =
@s_w_id10, @ol_qty = @ol_qty10
                        else   if @ol_number < @12SELECT @i_id = @i_id11,
@s_w_id = @s_w_id11, @ol_qty = @ol_qty11
                                else              SELECT @i_id = @i_id12, @s_w_id =
@s_w_id12, @ol_qty = @ol_qty12
                else if @ol_number < @15
                        if @ol_number < @14SELECT @i_id = @i_id13, @s_w_id =
@s_w_id13, @ol_qty = @ol_qty13
                        else              SELECT @i_id = @i_id14, @s_w_id =
@s_w_id14, @ol_qty = @ol_qty14
                else                      SELECT @i_id = @i_id15, @s_w_id =
@s_w_id15, @ol_qty = @ol_qty15

                OPEN c_no_is
                FETCH c_no_is INTO
                        @i_price, @i_name, @i_data,
                        @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
```

```
                    @s_quantity, @s_data,
                    @s_ytd, @s_order_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

            if (@@sqlstatus != 0) begin/* item not found */
                    SELECT @commit_flag = 0
                    select              /* Return to client */
                            NULL, NULL, NULL, NULL, NULL
                            break
            end

            if @d_id < @5
              if @d_id < @3
                    if @d_id < @2SELECT @s_dist = @s_dist_01
                    else        SELECT @s_dist = @s_dist_02
              else if @d_id < @4SELECT @s_dist = @s_dist_03
                    else        SELECT @s_dist = @s_dist_04
            else if @d_id < @7
                    if @d_id < @6SELECT @s_dist = @s_dist_05
                    else        SELECT @s_dist = @s_dist_06
              else if @d_id < @9
                     if @d_id < @8SELECT @s_dist = @s_dist_07
                     else       SELECT @s_dist = @s_dist_08
            else if @d_id < @10SELECT @s_dist = @s_dist_09
              else              SELECT @s_dist = @s_dist_10

            select @ol_qty_smallint = @ol_qty
            if @s_quantity >= @ol_qty_smallint + @ten
                    SELECT @s_quantity = @s_quantity -
@ol_qty_smallint
            else
                    SELECT @s_quantity = @s_quantity -
@ol_qty_smallint + 91

            UPDATE stock set
                    s_quantity= @s_quantity,
                    s_ytd       = @s_ytd + @ol_qty,
                    s_order_cnt= @s_order_cnt + @one
                    WHERE CURRENT OF c_no_is

            if (patindex("%ORIGINAL%", @i_data) > 0) and
               (patindex("%ORIGINAL%", @s_data) > 0)
                    SELECT @b_g = "B"
            else
                    SELECT @b_g = "G"

            SELECT @ol_amount = @ol_qty * @i_price
            INSERT INTO order_line (
                    ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
                    ol_supply_w_id, ol_delivery_d, ol_quantity,
                    ol_amount, ol_dist_info)
            VALUES (
                    @o_id, @d_id, @w_id, @ol_number+@o_ol_done,
@i_id,
                    @w_id, "19000101", @ol_qty_smallint,
                    @ol_amount, @s_dist)

            select                          /* Return to client */
                    @i_name,
                    @i_price,
                    @s_quantity,
                    @ol_amount,
                    @b_g
            CLOSE c_no_is
    end

    if (@o_ol_done + @o_ol_now >= @o_ol_cnt)/* Finish order */
    begin
            SELECT @o_entry_d = getdate()
            INSERT INTO orders (
                    o_id, o_c_id, o_d_id, o_w_id,
                    o_entry_d, o_carrier_id, o_ol_cnt, o_all_local)
            VALUES (
                    @o_id, @c_id, @d_id, @w_id,
                    @o_entry_d, -1, @o_ol_cnt, 1)
            INSERT INTO new_order (no_o_id, no_d_id, no_w_id)
            VALUES (@o_id, @d_id, @w_id)

            if (@o_ol_now >= @o_ol_cnt)
                    if (@commit_flag = @1)
                            commit transaction NO
                    else
                            rollback transaction NO
    end

    select          /* Return to client */
            @w_tax, @d_tax, @o_id, @c_last,
            @c_discount, @c_credit,
            convert(char(11),@o_entry_d,105) + convert(char(8),@o_entry_d,108)
end
go

if exists ( SELECT name FROM sysobjects WHERE name = 'neworder_remote')
    DROP PROC neworder_remote
go
CREATE PROC neworder_remote (
    @w_id       smallint,
    @d_id       tinyint,
    @c_id       int,
    @o_ol_cnttinyint,
    @o_ol_done      tinyint,
    @o_ol_now       tinyint,

    @i_idint, @s_w_idsmallint, @ol_qtytinyint,
    @i_id2int, @s_w_id2smallint, @ol_qty2tinyint,
    @i_id3int, @s_w_id3smallint, @ol_qty3tinyint,
    @i_id4int, @s_w_id4smallint, @ol_qty4tinyint,
    @i_id5int, @s_w_id5smallint, @ol_qty5tinyint,
    @i_id6int, @s_w_id6smallint, @ol_qty6tinyint,
    @i_id7int, @s_w_id7smallint, @ol_qty7tinyint,
    @i_id8int, @s_w_id8smallint, @ol_qty8tinyint,
    @i_id9int, @s_w_id9smallint, @ol_qty9tinyint,
    @i_id10int, @s_w_id10smallint, @ol_qty10tinyint,
    @i_id11int, @s_w_id11smallint, @ol_qty11tinyint,
    @i_id12int, @s_w_id12smallint, @ol_qty12tinyint,
    @i_id13int, @s_w_id13smallint, @ol_qty13tinyint,
    @i_id14int, @s_w_id14smallint, @ol_qty14tinyint,
    @i_id15int, @s_w_id15smallint, @ol_qty15tinyint,

    @o_id       int,
    @commit_flagtinyint
)
as
```

```
        @i_id15      int, @ol_qty15     tinyint,

        @o_id               int,
        @commit_flag    tinyint
)
as

declare
        @w_tax      real,          @d_tax      real,
        @c_last     char(16),@c_creditchar(2),
        @c_discountreal,

        @ol_amountreal,
        @i_pricereal,
        @i_name     char(24),@i_data char(50),

        @s_quantitysmallint,
        @s_ytd      int,           @s_order_cntint,
        @s_dist     char(24),@s_data char(50),
        @s_dist_01char(24),@s_dist_02char(24),
        @s_dist_03char(24),@s_dist_04char(24),
        @s_dist_05char(24),@s_dist_06char(24),
        @s_dist_07char(24),@s_dist_08char(24),
        @s_dist_09char(24),@s_dist_10char(24),

        @ol_numbertinyint,
        @o_entry_ddatetime,@b_gchar(1)

declare@0  tinyint,@1  tinyint,@2  tinyint,@3  tinyint,
        @4  tinyint,@5  tinyint,@6  tinyint,@7  tinyint,
        @8  tinyint,@9  tinyint,@10 tinyint,@11 tinyint,
        @12 tinyint,@13 tinyint,@14 tinyint,@15 tinyint,
        @one smallint,@ten smallint,@ol_qty_smallint smallint

declare c_no_wdc CURSOR FOR
        SELECTw_tax, d_tax, d_next_o_id,
                c_last, c_discount, c_credit
                FROMwarehouse HOLDLOCK,
                        district HOLDLOCK,
                        customer (index c_clu prefetch 2 lru) HOLDLOCK
                WHEREd_w_id= @w_id
                AND d_id  = @d_id
                AND w_id = d_w_id
                AND c_w_id= w_id
                AND c_d_id= d_id
                AND c_id  = @c_id
                FOR UPDATE OF d_next_o_id

declare c_no_is CURSOR FOR
        SELECTi_price, i_name, i_data,
                s_quantity, s_data,
                s_ytd, s_order_cnt,/* for update */
                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
                FROMitem HOLDLOCK,
                        stock HOLDLOCK
                WHEREs_w_id = @w_id
                AND s_i_id = i_id
                AND   i_id = @i_id
                FOR UPDATE OF s_quantity, s_ytd, s_order_cnt

begin
        select@0=0, @1=1, @2=2, @3=3, @4=4, @5=5, @6=6, @7=7, @8=8,
                @9=9, @10=10, @11=11, @12=12, @13=13, @14=14, @15=15,
                @one=1, @ten=10

        if (@o_ol_done = @0)/* Start a new order */
        begin
                if (@o_ol_now >= @o_ol_cnt)/* One call only */
                        begin transaction NO

                OPEN c_no_wdc
                FETCH c_no_wdc INTO
                        @w_tax, @d_tax, @o_id,
                        @c_last, @c_discount, @c_credit
                SELECT@commit_flag= 1
                UPDATEdistrict
                        SET   d_next_o_id = @o_id + 1
                        WHERE CURRENT OF c_no_wdc
                CLOSE c_no_wdc
        end

        SELECT @ol_number = @0
        while (@ol_number < @o_ol_cnt) begin
                SELECT @ol_number = @ol_number + @1

                if @ol_number < @5
                        if @ol_number < @3 begin
                                if @ol_number >= @2SELECT @i_id = @i_id2,  @ol_qty
= @ol_qty2
                        end
                        else   if @ol_number < @4SELECT @i_id = @i_id3,  @ol_qty
= @ol_qty3
                                else                SELECT @i_id = @i_id4,  @ol_qty =
@ol_qty4
                else if @ol_number < @9
                        if @ol_number < @7
                                if @ol_number < @6SELECT @i_id = @i_id5,  @ol_qty
= @ol_qty5
                                else                SELECT @i_id = @i_id6,  @ol_qty =
@ol_qty6
                        else   if @ol_number < @8SELECT @i_id = @i_id7,  @ol_qty
= @ol_qty7
                                else                SELECT @i_id = @i_id8,  @ol_qty =
@ol_qty8
                else if @ol_number < @13
                        if @ol_number < @11
                                if @ol_number < @10SELECT @i_id = @i_id9,  @ol_qty
= @ol_qty9
                                else                SELECT @i_id = @i_id10, @ol_qty =
@ol_qty10
                        else   if @ol_number < @12SELECT @i_id = @i_id11, @ol_qty
= @ol_qty11
                                else                SELECT @i_id = @i_id12, @ol_qty =
@ol_qty12
                else if @ol_number < @15
                        if @ol_number < @14SELECT @i_id = @i_id13, @ol_qty =
@ol_qty13
                        else                        SELECT @i_id = @i_id14, @ol_qty =
@ol_qty14
                else                                SELECT @i_id = @i_id15, @ol_qty =
@ol_qty15

                OPEN c_no_is
                FETCH c_no_is INTO
                        @i_price, @i_name, @i_data,
char          s_dist[25];

/*
** Variables for order table
*/

int           o_id;
DBTINYINT     o_d_id;
DBSMALLINT    o_w_id;
DBSMALLINT    o_c_id;
char          o_entry_d[31];
DBSMALLINT    o_carrier_id;
DBSMALLINT    o_ol_cnt, o_ol_now, o_ol_done;
DBTINYINT     o_all_local;

/*
** Variables for order_line
*/

int           ol_o_id;
DBTINYINTol_d_id;
DBSMALLINTol_w_id;
DBSMALLINTol_number;
DBINT         ol_i_id;
DBSMALLINTol_supply_w_id;
char          ol_delivery_d[31];
DBSMALLINTol_quantity;
DBFLT8        ol_amount;

/*
** Variables for new_order tble
*/

int           no_o_id;
DBTINYINTno_d_id;
DBSMALLINTno_w_id;

/*
** Variables for history table
*/

DBFLT8        h_amount;
char          h_date[20];

/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */



/* For NEWO:     */

struct newo_inf  newosp;
struct newo_inf  *neworder;
FBFR                 *newo_fbfr;
TPSVCINFO                *newo_rqst;
int                  newolen;
struct track_mods        mod_array[50],
                     *modptr = mod_array;


struct req_struct *delp;      /* Transaction message */
char             outbuf[1024];   /* Buffer for results file */
int              tx_count = 0;   /* Transaction counter */
FILE                *delfile;
TPSVCINFO        *del_rqst;


/* For ORDS:    */

struct ord_inf ordsp,
          *ordstat = &ordsp;
FBFR                *ordsbuf;   /* FML buffer for output carray */
int                 ordslen;    /* Size of FML buffer */
TPSVCINFO *ords_rqst;


/* For PAYM:*/

struct pay_inf *payment;        /* Input structure to payment_tx */
struct pay_inf paymsp;          /* Payment structure */
TPSVCINFO *paym_rqst;


/* For STOCK:   */

struct stock_inf *stocklevel;   /* Input to stocklevel transaction */
struct stock_inf stocksp;
TPSVCINFO *stock_rqst;



#!/bin/sh -f

# Stored procedure for TPC-C 3.0 on SQL Server 11.0 and later
# Copyright Sybase 1995

isql -e -Usa -P$PASSWORD <<EOF
use tpcc
go
if exists ( SELECT name FROM sysobjects WHERE name = 'neworder_local')
       DROP PROC neworder_local
go

CREATE PROC neworder_local (
       @w_id       smallint,
       @d_id       tinyint,
       @c_id       int,
       @o_ol_cnttinyint,
       @o_ol_done      tinyint,
       @o_ol_now       tinyint,

       @i_idint, @ol_qty        tinyint,
       @i_id2int, @ol_qty2     tinyint,
       @i_id3int, @ol_qty3     tinyint,
       @i_id4int, @ol_qty4     tinyint,
       @i_id5int, @ol_qty5     tinyint,
       @i_id6int, @ol_qty6     tinyint,
       @i_id7int, @ol_qty7     tinyint,
       @i_id8int, @ol_qty8     tinyint,
       @i_id9int, @ol_qty9     tinyint,
       @i_id10int, @ol_qty10  tinyint,
       @i_id11int, @ol_qty11  tinyint,
       @i_id12int, @ol_qty12  tinyint,
       @i_id13int, @ol_qty13  tinyint,
       @i_id14int, @ol_qty14  tinyint,
    case XACT_ORDS_ID:/* order_status_byid */
        userlog("cw=%d, cd=%d, c=%d\n", c_w_id, c_d_id, c_id);
        break;

    case XACT_ORDS_NAME:/* order_status_byname */
        userlog("cw=%d, cd=%d, l=%s\n", c_w_id, c_d_id, c_last);
        break;

    case XACT_DEL:/* delivery_qu */
        userlog("w=%d, carrier=%d\n", global_w_id, o_carrier_id);
        break;

    case XACT_STOCK:/* stock level */
        userlog("w=%d, d=%d, th=%d\n", global_w_id, global_d_id,
threshold);
        break;

    case XACT_BKEND:/* delivery */
        userlog("w=%d, d=%d, carrier=%d, tx_count=%d\n",
                        global_w_id, global_d_id, o_carrier_id, tx_count);
        break;

    default:
        userlog("Unknown xact_type = %d\n", xact_type);
    }
}

/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */



XCTION func_array[XCTION_COUNT+1] =
    {
        {"new_order"},
        {"payment_byid"},
        {"payment_byname"},
        {"order_status_byid"},
        {"order_status_byname"},
        {"delivery_qu"},
        {"stock_level"},
        {"delivery"},
        {"NULL"}
    };

int             rollback_pct;
int             lines_per_call = 15;
char            b_g[2];
DBFLT8          total_amount;
DBTINYINT commit_flag;
int             xact_type, prev_xact_type = -9999;
int             deadlock;
int             bad_items;
int             max_ware;
char            *db_name = "tpcc";
RETCODE code;

DBSMALLINTglobal_w_id;
DBTINYINTglobal_d_id;
ORDER_LINEol[15];/* XXX: should be 16, or 15 ?? */

/*
** Variables for the customer table
*/

DBINT       c_id;
DBTINYINTc_d_id;
DBSMALLINTc_w_id;
char        c_first[17];
char        c_middle[3];
char        c_last[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_since[31];
char        c_credit[3];
DBFLT8      c_credit_lim;
DBREAL      c_discount;
DBFLT8      c_balance;
char        c_data[201];

/*
** Variables for warehouse
*/

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];
DBREAL      w_tax;

/*
** Variables for district
*/

DBTINYINT       d_id;
DBSMALLINTd_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];
DBREAL  d_tax;

/*
** Variables for item table
*/

int         i_id;
DBFLT8      i_price;
char        i_name[25];

/*
** Variables for the stock table
*/

DBSMALLINTs_quantity;
DBSMALLINTthreshold;
DBINT       low_count;
```

```
            fwrite(outbuf, strlen(outbuf), 1, delfile);
            fflush(delfile);
            tpreturn(TPFAIL, 0, (char *)del_rqst->data, del_rqst->len, 0);
        }
    }
}

int
delivery_body()
{

    deadlock = 0;
    dbrpcinit(dbproc, "delivery", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &global_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &global_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &o_carrier_id);
    if (dbrpcsend(dbproc) != SUCCEED) return TRUE;
    if (dbsqlok(dbproc) != SUCCEED) return TRUE;

    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;

    dbbind(dbproc, 1, INTBIND,      0,     &o_id);
    dbbind(dbproc, 2, NTBSTRINGBIND, sizeof(ol_delivery_d),
ol_delivery_d);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;

    if (o_id == NULL)
        sprintf(outbuf+strlen(outbuf),
            "Delivery for District %d skipped\n", global_d_id);
    else
        sprintf(outbuf+strlen(outbuf),
            "Delivered order %d for district %d, warehouse %d, carrier
%d\n",
            o_id, global_d_id, global_w_id, o_carrier_id);

    if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;
    return FALSE;
}

void
stock_level_rpc()
{
    int try;

    for (try = 0; try < MaxTries; try ++)
    {
        if (try > 0) display_xction("Repeating");

        if (stock_level_body() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        break;
    }

    if (try >= MaxTries)
    {
        display_xction("MaxTries Failed");
        tpreturn(TPFAIL, 0, (char *)stock_rqst->data, stock_rqst->len, 0);
    }
}

int
stock_level_body()
{
    deadlock = 0;
    dbrpcinit(dbproc, "stock_level", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &global_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &global_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &threshold);
    if (dbrpcsend(dbproc) != SUCCEED) return TRUE;
    if (dbsqlok(dbproc) != SUCCEED) return TRUE;

    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;
    dbbind(dbproc, 1, INTBIND,  0, &low_count);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;
    if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;

    stocklevel->low_stock = low_count;

    return FALSE;
}

void ins_rpc()
{
    dbfcmd(dbproc,"insert into foo values(%d, 'kjhkjhkjhkjhkjh')", global_w_id);
    dbsqlexec(dbproc);
    dbresults(dbproc);
}

void
sleep_before_retry()
{
    sleep(1);
}

void
display_xction(msg)
char *msg;
{
    int i;
    userlog("%s %s ", msg, func_array[xact_type].name);

    switch(xact_type)
    {

    case XACT_NEWO:/* new_order */
        userlog("w=%d, d=%d, c=%d, %d lines: \n[",
                    global_w_id, global_d_id, c_id, o_ol_cnt);
        for (i=0; i<(int)o_ol_cnt; i++)
            userlog(" %d", ol[i].i_id);
        userlog("]\n");
        break;

    case XACT_PAYM_ID:/* payment_byid */
        userlog("w=%d/%d, d=%d/%d, c=%d\n",
                    global_w_id, c_w_id, global_d_id, c_d_id, c_id);
        break;

    case XACT_PAYM_NAME:/* payment_byname */
        userlog("w=%d/%d, d=%d/%d, l=%s\n",
                    global_w_id, c_w_id, global_d_id, c_d_id, c_last);
        break;
    for (try=0; try<MaxTries; try++)
    {
        if (try>0) display_xction("Repeating");

        if (order_status_byname_begin() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        if (order_status_end() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        break;
    }

    if (try >= MaxTries) {
            display_xction("MaxTries Failed");
            tpreturn(TPFAIL, 0, (char *)ords_rqst->data, ords_rqst->len, 0);
    }
}

int
order_status_byname_begin()
{
    deadlock = 0;
    dbrpcinit(dbproc, "order_status_byname", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &c_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &c_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBCHAR, -1, strlen(c_last), c_last);
    return (dbrpcsend(dbproc) == SUCCEED ? FALSE : TRUE);
}

int
order_status_end()
{
    int         count;

    if (dbsqlok(dbproc) != SUCCEED) return TRUE;

    if (dbresults(dbproc) != SUCCEED || deadlock)
        return TRUE;
    else {
        dbbind(dbproc, 1, SMALLBIND,     0,     &ol_supply_w_id);
        dbbind(dbproc, 2, INTBIND,       0,     &ol_i_id);
        dbbind(dbproc, 3, SMALLBIND,     0,     &ol_quantity);
        dbbind(dbproc, 4, FLT8BIND,      0,     &ol_amount);
        dbbind(dbproc, 5, NTBSTRINGBIND, sizeof(ol_delivery_d),
ol_delivery_d);

        count = 0;
        while ((code = dbnextrow(dbproc)) == REG_ROW && !deadlock)
        {
            /*
            ** Print order_line information on RTE
            */
                ordstat->o_items[count].ol_supply_w_id = ol_supply_w_id;
                ordstat->o_items[count].ol_i_id = ol_i_id;
                ordstat->o_items[count].ol_quantity = ol_quantity;
                ordstat->o_items[count].ol_amount = ol_amount;
                strcpy(ordstat->o_items[count].ol_delivery_d, ol_delivery_d);
                count++;
        }
        ordstat->item_cnt = count;

        if (code != NO_MORE_ROWS || deadlock) return TRUE;
    }

    if (dbresults(dbproc) != SUCCEED || deadlock)
        return TRUE;
    else
    {
        dbbind(dbproc, 1, INTBIND,      0,     &c_id);
        dbbind(dbproc, 2, NTBSTRINGBIND, sizeof(c_last), c_last);
        dbbind(dbproc, 3, NTBSTRINGBIND, sizeof(c_first), c_first);
        dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(c_middle), c_middle);
        dbbind(dbproc, 5, FLT8BIND,      0,     &c_balance);
        dbbind(dbproc, 6, INTBIND,       0,     &o_id);
        dbbind(dbproc, 7, NTBSTRINGBIND, sizeof(o_entry_d), o_entry_d);
        dbbind(dbproc, 8, SMALLBIND,     0,     &o_carrier_id);
        if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;
        if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;

        strcpy(ordstat->c_first, c_first);
        strcpy(ordstat->c_middle, c_middle);
        strcpy(ordstat->c_last, c_last);
        ordstat->c_balance = c_balance;
        ordstat->o_id = (int)o_id;
        strcpy(ordstat->o_entry_d, o_entry_d);
        ordstat->o_carrier_id = o_carrier_id;

    }

    return FALSE;
}

void
delivery_rpc()
{
/*
 Called by delivery processes.
*/
    int try;


    for (global_d_id = 1; global_d_id < (DBTINYINT)11; global_d_id++)
    {
        for (try = 0; try < MaxTries; try++)
        {
            if (try > 0) display_xction("Repeating");

            if (delivery_body() == TRUE)
            {
                dbcancel(dbproc);
                sleep_before_retry();
                continue;
            }
            break;
        }
        if (try >= MaxTries)
        {
            display_xction("MaxTries Failed");
    if (try >= MaxTries)
    {
            display_xction("MaxTries Failed");
            tpreturn(TPFAIL, 0, (char *)paym_rqst->data, paym_rqst->len,
0);
    }
}

int
payment_byname_begin()
{
    deadlock = 0;
    dbrpcinit(dbproc, "payment_byname", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &global_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &c_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBFLT8, -1, -1, &h_amount);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &global_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &c_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBCHAR, -1, strlen(c_last), c_last);
    return (dbrpcsend(dbproc) == SUCCEED ? FALSE : TRUE);
}

int
payment_end()
{


    if (dbsqlok(dbproc) != SUCCEED) return TRUE;
    if (dbresults(dbproc) != SUCCEED || deadlock)
            return TRUE;
    else
    {
        dbbind(dbproc,  1, INTBIND,     0, &c_id);
        dbbind(dbproc,  2, NTBSTRINGBIND, sizeof(c_last), c_last);
        dbbind(dbproc,  3, NTBSTRINGBIND, sizeof(h_date), h_date);
        dbbind(dbproc,  4, NTBSTRINGBIND, sizeof(w_street_1),
w_street_1);
        dbbind(dbproc,  5, NTBSTRINGBIND, sizeof(w_street_2),
w_street_2);
        dbbind(dbproc,  6, NTBSTRINGBIND, sizeof(w_city), w_city);
        dbbind(dbproc,  7, NTBSTRINGBIND, sizeof(w_state), w_state);
        dbbind(dbproc,  8, NTBSTRINGBIND, sizeof(w_zip), w_zip);

        dbbind(dbproc,  9, NTBSTRINGBIND, sizeof(d_street_1),
d_street_1);
        dbbind(dbproc, 10, NTBSTRINGBIND, sizeof(d_street_2),
d_street_2);
        dbbind(dbproc, 11, NTBSTRINGBIND, sizeof(d_city), d_city);
        dbbind(dbproc, 12, NTBSTRINGBIND, sizeof(d_state), d_state);
        dbbind(dbproc, 13, NTBSTRINGBIND, sizeof(d_zip), d_zip);

        dbbind(dbproc, 14, NTBSTRINGBIND, sizeof(c_first), c_first);
        dbbind(dbproc, 15, NTBSTRINGBIND, sizeof(c_middle),
c_middle);
        dbbind(dbproc, 16, NTBSTRINGBIND, sizeof(c_street_1),
c_street_1);
        dbbind(dbproc, 17, NTBSTRINGBIND, sizeof(c_street_2),
c_street_2);
        dbbind(dbproc, 18, NTBSTRINGBIND, sizeof(c_city), c_city);
        dbbind(dbproc, 19, NTBSTRINGBIND, sizeof(c_state), c_state);
        dbbind(dbproc, 20, NTBSTRINGBIND, sizeof(c_zip), c_zip);
        dbbind(dbproc, 21, NTBSTRINGBIND, sizeof(c_phone), c_phone);
        dbbind(dbproc, 22, NTBSTRINGBIND, sizeof(c_since), c_since);
        dbbind(dbproc, 23, NTBSTRINGBIND, sizeof(c_credit), c_credit);
        dbbind(dbproc, 24, FLT8BIND,      0, &c_credit_lim);
        dbbind(dbproc, 25, REALBIND,      0, &c_discount);
        dbbind(dbproc, 26, FLT8BIND,      0, &c_balance);
        dbbind(dbproc, 27, NTBSTRINGBIND, sizeof(c_data), c_data);
        if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;

        if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;
    }
    return FALSE;
}

void
order_status_byid_rpc()
{
    int try;


    for (try=0; try<MaxTries; try++)
    {
        if (try>0) display_xction("Repeating");

        if (order_status_byid_begin() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        if (order_status_end() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        break;
    }

    if (try >= MaxTries)
    {
        display_xction("MaxTries Failed");
        tpreturn(TPFAIL, 0, (char *)ords_rqst->data, ords_rqst->len, 0);
    }
}

int
order_status_byid_begin()
{
    deadlock = 0;


    dbrpcinit(dbproc, "order_status_byid", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &c_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &c_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, &c_id);
    return (dbrpcsend(dbproc) == SUCCEED ? FALSE : TRUE);
}

void
order_status_byname_rpc()
{
    int try;
```

```
            {

                strcpy(neworder->status, “Item number is not valid”);
                commit_flag = FALSE;
            }
            if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;
        }
        if (dbhasretstat(dbproc))
        {
            if ((retcode = dbretstatus(dbproc)) == -3)
            {
                deadlock = 1;
                display_xction(“Deadlock victim:”);
            }
            else if (retcode<0)
            {
                userlog(“Unknown return status %d:”, retcode);
                display_xction(““);
            }
            return TRUE;
        }

        cur_ip = &neworder->n_items[i];
        strcpy(cur_ip->i_name, i_name);
        cur_ip->i_price = i_price;
        cur_ip->s_quantity = s_quantity;
        strcpy(cur_ip->brand, b_g);
        cur_ip->ol_amount = ol_amount;

        total_amount += ol_amount;
    }


    if (dbresults(dbproc) != SUCCEED || deadlock)
        {
        return TRUE;
    }

    dbbind(dbproc, 1, REALBIND, 0,&w_tax);
    dbbind(dbproc, 2, REALBIND, 0, &d_tax);
    dbbind(dbproc, 3, INTBIND,  0, &o_id);
    dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(c_last),c_last);
    dbbind(dbproc, 5, REALBIND,     0,&c_discount);
    dbbind(dbproc, 6, NTBSTRINGBIND, sizeof(c_credit),c_credit);
    dbbind(dbproc, 7, NTBSTRINGBIND, sizeof(o_entry_d),o_entry_d);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;

    neworder->w_tax = w_tax;
    neworder->d_tax = d_tax;
    neworder->o_id = o_id;
    strcpy(neworder->c_last, c_last);
    neworder->c_discount = c_discount;
    strcpy(neworder->c_credit, c_credit);
    strcpy(neworder->o_entry_d, o_entry_d);

    return FALSE;
}

void
payment_byid_rpc()
{
    int try;

    for (try=0; try<MaxTries; try++)
    {
        if (try>0) display_xction(“Repeating”);

        if (payment_byid_begin() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        if (payment_end() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        break;
    }

    if (try >= MaxTries)
    {
            display_xction(“MaxTries Failed”);
            tpreturn(TPFAIL, 0, (char *)paym_rqst->data, paym_rqst->len, 0);
    }
}

int
payment_byid_begin()
{
    deadlock = 0;
    dbrpcinit(dbproc, “payment_byid”, 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &global_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &c_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBFLT8, -1, -1, &h_amount);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &global_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &c_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, &c_id);
    return (dbrpcsend(dbproc) == SUCCEED ? FALSE : TRUE);
}

void
payment_byname_rpc()
{
    int try;

    for (try=0; try<MaxTries; try++)
    {
        if (try>0) display_xction(“Repeating”);

        if (payment_byname_begin() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        if (payment_end() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        break;
    }
```

```
 /* Tuxedo include files */
#include "atmi.h"
#include "userlog.h"
#include "fml.h"
#include "mods.h"
#include "Usysflds.h"

#include "tpcc_tux_forms.h"
#include "tpcc_tux_forms_var.c"


void
new_order_rpc()
{
    int        try;


    for (try=0; try<MaxTries; try++)
    {
        if (try > 0) display_xction("Repeating");

        commit_flag = TRUE;
        deadlock = 0;

        if (o_ol_cnt > lines_per_call)
        {
            dbcmd(dbproc, "begin transaction NO");
            if (dbsqlexec(dbproc) != SUCCEED) goto Failed;
            while ((code = dbresults(dbproc)) == SUCCEED && !deadlock)
                dbcanquery(dbproc);
            if (code == FAIL || deadlock) goto Failed;
        }

        total_amount = 0;
        for (o_ol_done=0; o_ol_done < o_ol_cnt; o_ol_done +=
lines_per_call)
        {
            o_ol_now = smaller(o_ol_cnt-o_ol_done,
(DBTINYINT)lines_per_call);
            if (new_order_body() == TRUE) goto Failed;

        }
        total_amount *= (1+w_tax+d_tax)*(1-c_discount);

        if (o_ol_cnt > lines_per_call)
        {

            dbfcmd(dbproc, "%s transaction NO",
                    commit_flag == TRUE ? "commit" : "rollback");
            dbsqlexec(dbproc);
            while ((code = dbresults(dbproc)) == SUCCEED && !deadlock)
                dbcanquery(dbproc);
            if (code == FAIL || deadlock) goto Retry;
        }
        break;

Failed:
        dbcancel(dbproc);
        if (o_ol_cnt > lines_per_call)
        {
            dbcmd(dbproc, "rollback transaction NO");
            dbsqlexec(dbproc);
            while (dbresults(dbproc) == SUCCEED)
                dbcanquery(dbproc);
        }
Retry:
        dbcancel(dbproc);
        sleep_before_retry();
    }

    if (try >= MaxTries) {
        display_xction("MaxTries Failed");
        tpreturn(TPFAIL, 0, (char *)newo_rqst->data, newo_rqst->len, 0);
    }
}


int
new_order_body()
{
    int i,j;
    DBINT retcode;
    struct items_inf *cur_ip;       /* Pointer to current item */

    deadlock = 0;
    if (o_all_local)
        dbrpcinit(dbproc, "neworder_local", 0);
    else
        dbrpcinit(dbproc, "neworder_remote", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2,  -1, -1, &global_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1,  -1, -1, &global_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT4,  -1, -1, &c_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT2,  -1, -1, &o_ol_cnt);
    dbrpcparam(dbproc, NULL, 0, SYBINT2,  -1, -1, &o_ol_done);
    dbrpcparam(dbproc, NULL, 0, SYBINT2,  -1, -1, &o_ol_now);

    for(i = o_ol_done; i < (int)o_ol_done+lines_per_call; i++)
    {
        dbrpcparam(dbproc, NULL, 0, SYBINT4,  -1, -1, &ol[i].i_id);
        if (!o_all_local)
                dbrpcparam(dbproc, NULL, 0, SYBINT2,  -1, -1, &ol[i].supply_w_id);
                dbrpcparam(dbproc, NULL, 0, SYBINT2,  -1, -1, &ol[i].quantity);

    }

    dbrpcparam(dbproc, NULL, 0, SYBINT4,  -1, -1, &o_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1,  -1, -1, &commit_flag);

    if (dbrpcsend(dbproc) != SUCCEED) return TRUE;
    if (dbsqlok(dbproc)   != SUCCEED) return TRUE;

    for (i = o_ol_done; i < (int)(o_ol_done+o_ol_now); i++)
    {
        if (dbresults(dbproc) != SUCCEED || deadlock)
            return TRUE;
        else
        {
            dbbind(dbproc, 1, NTBSTRINGBIND,  sizeof(i_name), i_name);
            dbbind(dbproc, 2, FLT8BIND,       0, &i_price);
            dbbind(dbproc, 3, SMALLBIND,      0, &s_quantity);
            dbbind(dbproc, 4, FLT8BIND,       0, &ol_amount);
            dbbind(dbproc, 5, NTBSTRINGBIND,  sizeof(b_g), b_g);
            if (dbnextrow(dbproc) != REG_ROW) return TRUE;


            if(*i_name == '\0')
        if (llen) {/* If user entered any input */
                *moneyptr = ‘\0’;
                *buf = atof(money);/* Result amount */
        }
        return(0);
}

/*
 * Function: get_string
 * Get a string variable
 * Returns: Terminating character
 */
int
get_string(buf, len, field_no)
char *buf;
int len, field_no;
{
        int remlen, term, i, llen = len;
        char *p1,*p2;

        p1 = (char *)field_array[field_no];
        for (i=0;i<len;i++)
        {
        buf[i] = *p1;
                p1++;
        }
        buf[len] = ‘\0’;/* Result string */
        return(0);
}

/*.
** (c) Copyright Sybase, Inc. 1991
** All rights reserved
**
** Version 94.04.26
** Modified by Keng-Tai Ko [05/19/94]
*/
#include <stdio.h>
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
#include “SYB_tpcc.h”

#define     CONTEXT_SET         5701
#define     LANGUAGE_SET        5703
#define     CHARACTER_SET       5704
#define     ABORT_ERROR         6104

/* Tuxedo include files */
#include “atmi.h”
#include “userlog.h”

int
err_handler(dbproc, severity, errno, oserr)
        DBPROCESS *dbproc;
        int severity;
        int errno;
        int oserr;
{
        userlog(“DB-LIBRARY Error %d:”, errno);
        display_xction(dberrstr(errno));

        if (oserr != DBNOERR)
                {
                userlog(“O/S Error: “);
                display_xction(dboserrstr(oserr));
                }

        /* exit on any error */
        exit(-100);
}



int
msg_handler(dbproc,msgno,msgstate,severity,msgtext,servername,procname,
line)
        DBPROCESS*dbproc;
        int             msgno;
        int             msgstate;
        int             severity;
        char            *msgtext;
        char             *servername;
        char             *procname;
        int             line;
{
        if (msgno == CONTEXT_SET ||
            msgno == LANGUAGE_SET ||
            msgno == CHARACTER_SET)
                return(SUCCEED);

        if (msgno == ABORT_ERROR)
                return(SUCCEED);

        if (msgno == 1205)
        {
                display_xction(msgtext);
                deadlock = 1;
                return(SUCCEED);
        }
        else {
                userlog(“msg no %d - %s\n”, msgno, msgtext);
                userlog(“xact_type: %d deadlock= %d\n”, xact_type, deadlock);
                if (msgno == 0)
                        return(SUCCEED);
                else
                        return(FAIL);
        }
}

/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */


#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>
#include <time.h>
#include <sybfront.h>
#include <sybdb.h>

#include “SYB_tpcc.h”
#include “SYB_driver.h”
#include “SYB_rpc_var.c”
```

```
                {
                        if (!was_in_field)
                                field_array[current_max_field] = input_buff_ptr;
                        was_in_field = 1;
                }
                input_buff_ptr++;
        }
}

print_error_on_field()
{
        int ret_code;

        ret_code = write(1,send_beep,strlen(send_beep));
}


/*
 * Function: get_char
 * Get a single char
 */
int
get_char(buf, field_no)
char *buf;
int field_no;
{
        int term, i, llen = 0, remlen, result, end, oldlen = 0;
        char chbuf[20], outbuf[20], *p;


        *buf = *field_array[field_no];

        return(0);
}
/*
 * Function: get_integer
 * Get an integer variable
 */
int
get_integer(buf, len,field_no)
int *buf;
int len, field_no;
{
        int term, i, llen = 0, remlen, result, end, oldlen = 0;
        char chbuf[20], outbuf[20], *p;

        oldlen = llen;
        llen = len; result = 0; end = 0;

        p = field_array[field_no];


        /* Now validate buf to see if it is an integer */
        if (oldlen > llen)/* If leftover input from previous edit, use it */
                llen = oldlen;
        for (i=0; i < llen; i++, p++) {
                if ( *p >= '0' && *p <= '9' && !end )
                        result = result * 10 + ( *p - '0');
                else if ( *p == ' ' )
                        end = 1;     /* Allow blanks only at end of input */
                else {
                        print_error_on_field(field_no);
                        return(INVALID_INTEGER);
                }
        }
        if (llen) {     /* If user entered any input */
                *buf = result;          /* Result integer */
        }
        return(0);
}

/*
 * Function: get_amount
 * Get an amount variable
 */
int
get_amount(buf, len ,field_no)
double *buf;
int len, field_no;
{
        int term, i, remlen, llen = 0, oldlen = 0, dotfound, num_afterdots;
        char chbuf[20], *p;
        char money[20], *moneyptr;
        char outbuf[20];
        int end = 0;

        oldlen = llen;/* Length from previous edit */
        llen = len; end = 0;
        moneyptr = money;
        p = field_array[field_no];

        /* Now validate buf to see if it is an amount */
        dotfound = 0;
        num_afterdots = 0;
        if (oldlen > llen)/* Use leftover from previous edit */
                llen = oldlen;
        for (i=0; i < llen; i++, p++) {
                if ( *p >= '0' && *p <= '9'  && !end ) {
                        if (dotfound) {/* fractional part */
                                if (num_afterdots >= 2) {
                        print_error_on_field(field_no);
                        return(INVALID_INTEGER);
                                }
                                num_afterdots++;
                        }
                        *moneyptr++ = *p;
                }
                else if ( *p == '.'  && !end ) {
                        if ( ! dotfound )
                                dotfound = 1;
                        else {
                           print_error_on_field(field_no);
                           return(INVALID_INTEGER);
                        }
                        *moneyptr++ = *p;
                }
                else if ( *p == ' ' )
                        end = 1;     /* Allow blanks only at end */
                else {
                     print_error_on_field(field_no);
                     return(INVALID_INTEGER);
                }
        }

        /* If length is less than requested, redisplay input
         * right-justified */
```

```
#include <stdio.h>
#include <syslog.h>
#include <errno.h>
#include "tpcc_term.h"

#define INVALID_INTEGER 1

extern double atof();
extern int w_id, d_id;

char *field_array[60];
int current_max_field;
int current_field;
char screen_buffer[3000];

unsigned char init_terminal[] = {
27,'x','0','\t',0x80,
27,'x','1',0x80,0x80,
27,'x','4','\r',0x80,0 };

unsigned char term_out[] = { 27,'V',0};


char send_beep [] = {7,0};
char disable_screen_buffer [] = {27,'&',27,')',0};

unsigned char first_screen[] = {
27,'*',     /* Clear to nulls */
27,',',     /* Clear to protected */
27,0x22,
27,')','W','a','r','e','h','o','u','s','e',' ',27,'(',' ',' ',' ',' ',
27,')', 'D','i','s','t','r','i','c','t',' ',27,'(',' ',' ',
27,')',
27,'&',
27,'N',
27,'(',
27,'0','1','S',
27,'B', 0 };



init_scr()
{
        int ret_code;
        int terminate = 0;


        ret_code = write(1,init_terminal,strlen(init_terminal));
        ret_code = write(1,first_screen,strlen(first_screen));


        current_field = 0;
        /* Display INIT screen and get user input for w_id, d_id */
        while (!terminate) {
                get_input(current_field);
                if (get_integer(&w_id, 4,current_field ) == 0) {
                        current_field++;
                        if (get_integer(&d_id, 2,current_field) == 0)
                                terminate = 1;
                }
        }
}
```

```
restore_scr()
{
}

/* Disable the screen and wait for the user to hit the send key */
wait_for_send()
{

        write(1,disable_screen_buffer,strlen(disable_screen_buffer));
        get_input(0);


}



/*
 * INPUT ROUTINES
 */

int
get_input(field_no)
int field_no;
{
        /* Get Input string */

        int i = 0, ch;
        int terminate=0,was_in_field = 0 ;
        char *input_buff_ptr;
        char tab_buffer[256],*tab_buffer_ptr;
        int number_char_returned,residue;

        input_buff_ptr = screen_buffer;
        residue = sizeof(screen_buffer);
        tab_buffer_ptr = tab_buffer;
        for (i=0;i<field_no+1;i++)
                *tab_buffer_ptr++ = '\t';
        *tab_buffer_ptr = 0;
        write(1,tab_buffer,strlen(tab_buffer));
        while (!terminate)
        {
                number_char_returned = read(0,input_buff_ptr,residue);

                input_buff_ptr += number_char_returned;
                residue -=number_char_returned;
                if ((*(input_buff_ptr - 1) == '\r') ||
                        (*(input_buff_ptr - 2) == '\r'))
                        terminate = 1;
        }
        input_buff_ptr = screen_buffer;
        current_max_field = 0;
        was_in_field = 0;
        while (*input_buff_ptr != '\r')
        {
                if (*input_buff_ptr == '\t')
                {
                        *input_buff_ptr = 0;
                        if (was_in_field)
                                current_max_field++;
                        was_in_field = 0;
                }
                else
```

```
ordstat_tx()
{
        long olen;

        *ordsp = ordstat;/* Copy structure to Tuxedo buffer */

        if (tpcall(ords_service, (char *)ordsp, sizeof(struct ord_struct),
                (char **)&ordsp, &olen, TPSIGRSTRT|TPNOTIME) == -1) {
                syslog(LOG_ERR, "ords tpcall failed. tperrno = %d",
tperrno);
                if (tperrno == TPEOS)
                        syslog(LOG_ERR, "Uunixerr = %d", Uunixerr);
                return(TXERRCODE);
        }
        ordstat = *ordsp;
        return(ordsp->item_cnt);
}

 /*
 * Payment Tuxedo client
 * Author : Shanti S
 * Date   : 8/03/93
 *
 */
#include <stdio.h>
#include <syslog.h>

#include "tpcc_client.h"

 /* Tuxedo includes */
#include "atmi.h"
#include "Uunix.h"

struct pay_struct *paymp;


 /*
 * Function: init payment
 * Init. Tuxedo
 */
init_paym_tx()
{
        if ((paymp = (struct pay_struct *)tpalloc("CARRAY", NULL,
sizeof(struct pay_struct))) == NULL) {
                syslog(LOG_ERR, "paym tpalloc failed. tperrno = %d",
tperrno);
                tpterm();
                cleanup(1);
        }
}


payment_tx()
{
        long olen;

        *paymp = payment;/* Copy structure to Tuxedo buffer */

        if (tpcall(paym_service, (char *)paymp, sizeof(struct pay_struct),
                (char **)&paymp, &olen, TPSIGRSTRT|TPNOTIME) == -1)
{
                syslog(LOG_ERR, "paym tpcall failed. tperrno = %d",
tperrno);
                if (tperrno == TPEOS)
                        syslog(LOG_ERR, "Uunixerr = %d", Uunixerr);
                return(TXERRCODE);
        }
        payment = *paymp;/* Copy results back */
        return(0);
}

 /*
 * Stocklevel client Tuxedo code
 * Author : Shanti S
 * Date   : 8/04/93
 *
 */

#include <stdio.h>
#include <syslog.h>

#include "tpcc_client.h"

 /* Tuxedo */
#include "atmi.h"
#include "Uunix.h"

struct stock_struct *stockp;
 /*
 * Function: init stocktat
 * Init. Tuxedo
 */
init_stock_tx()
{
        if ((stockp = (struct stock_struct *)tpalloc("CARRAY", NULL, sizeof(struct
stock_struct))) == NULL) {
                syslog(LOG_ERR, "stock tpalloc failed. tperrno = %d", tperrno);
                tpterm();
                cleanup(1);
        }
}


int
stocklevel_tx()
{
        long olen;

        *stockp = stocklevel;/* Copy structure to Tuxedo buffer */

        if (tpcall(stock_service, (char *)stockp, sizeof(struct stock_struct),
                (char **)&stockp, &olen, TPSIGRSTRT|TPNOTIME) == -1) {
                syslog(LOG_ERR, "stock tpcall failed. tperrno = %d", tperrno);
                if (tperrno == TPEOS)
                        syslog(LOG_ERR, "Uunixerr = %d", Uunixerr);
                return(TXERRCODE);
        }
        stocklevel.low_stock = stockp->low_stock;
        return(0);
}

 /*
 * (c) Copyright Sun Microsystems Inc. 1994
 * Written: 15/10/94
 *
 */
	/* Copy structure to Tuxedo buffer */
	delp->w_id = delivery.w_id;
	delp->o_carrier_id = delivery.o_carrier_id;
	time(&delp->qtime);


	if (tpacall(del_service, (char *)delp, sizeof(struct req_struct),
		TPSIGRSTRT | TPNOREPLY) == -1) {
		syslog(LOG_ERR, "del tpacall failed. tperrno = %d", tperrno);
		if (tperrno == TPEOS)
			syslog(LOG_ERR, "Uunixerr = %d", Uunixerr);
		return(TXERRCODE);
	}
	strcpy(delivery.status, "Delivery has been queued");
	return(0);
}

/*
 * Neworder Tuxedo client
 * Author : Shanti S
 * Date   : 8/03/93
 *
 */
#include <stdio.h>
#include <syslog.h>

#include "tpcc_client.h"

/* Tuxedo includes */
#include "atmi.h"
#include "Uunix.h"

struct no_struct *newop;


/*
 * Function: init neworder
 * This is the first function called by do_init, hence we initialize
 * Tuxedo here using 'tpinit'
 * Init. Tuxedo
 */
init_newo_tx()
{
	if (tpinit(NULL) == -1) {
		syslog(LOG_ERR, "tpinit failed. tperrno = %d", tperrno);
		if (tperrno == TPEOS)
			syslog(LOG_ERR, "Unix system call that failed = %d",
Uunixerr);
		cleanup(1);
	}
	if ((newop = (struct no_struct *)tpalloc("CARRAY", NULL,
sizeof(struct no_struct))) == NULL) {
		syslog(LOG_ERR, "newo tpalloc failed. tperrno = %d",
tperrno);
		tpterm();
		cleanup(1);
	}
}


/*
 * This function is called by cleanup to exit from Tuxedo
 */
restore_tx()
{
	tpterm();
}



neworder_tx(linecnt)
int linecnt;/* Number of lines on order */
{
	long olen;

	*newop = neworder;/* Copy structure to Tuxedo buffer */
	newop->o_ol_cnt = linecnt;

	if (tpcall(newo_service, (char *)newop, sizeof(struct no_struct),
		(char **)&newop, &olen, TPSIGRSTRT | TPNOTIME) == -1) {
		if (tperrno == TPEOS) {
		/* Don't log errors always, as outofspace errors can pile up */
		syslog(LOG_ERR, "newo tpcall failed.tperrno = %d, Uunixerr =
%d",
		tperrno, Uunixerr);
		}
		return(TXERRCODE);
	}
	neworder = *newop;/* Copy results back */
	return(0);
}

/*
 * Order-status client Tuxedo code
 * Author : Shanti S
 * Date   : 8/02/93
 *
 */

#include <stdio.h>
#include <syslog.h>

#include "tpcc_client.h"

/* Tuxedo */
#include "atmi.h"
#include "Uunix.h"

struct ord_struct *ordsp;
/*
 * Function: init ordstat
 * Init. Tuxedo
 */
init_ords_tx()
{
	if ((ordsp = (struct ord_struct *)tpalloc("CARRAY", NULL, sizeof(struct
ord_struct))) == NULL) {
		syslog(LOG_ERR, "ords tpalloc failed. tperrno = %d", tperrno);
		tpterm();
		cleanup(1);
	}
}


int
```

```
                payment.c_data_4);
        ret_code = write(1,display_buffer,strlen(display_buffer));
        sprintf(display_buffer,(char *)term_sequence);
        ret_code = write(1,display_buffer,strlen(display_buffer));
        wait_for_send();
}


put_ordstat(count)
int count;
{
        int ret_code;
        struct fld *nextp = ordstat_fld;
        struct ord_itm_struct *valp = ordstat.o_items;
        int i, cur_y;


        sprintf(display_buffer,(char *)ordstat_header_output_screen,
                ordstat.d_id,
                ordstat.c_id,
                ordstat.c_first,   /* customer name */
                ordstat.c_middle,
                ordstat.c_last,
                ordstat.c_balance,
                ordstat.o_id,         /* order-id */
                ordstat.o_entry_d,         /* entry date */
                ordstat.o_carrier_id);
        ret_code = write(1,display_buffer,strlen(display_buffer));

        for (i = 0; i < count; i++, valp++) {
        sprintf(display_buffer,(char *)order_line_output_screen,
                valp->ol_supply_w_id,
                valp->ol_i_id,
                valp->ol_quantity,
                valp->ol_amount,
                valp->ol_delivery_d);
        ret_code = write(1,display_buffer,strlen(display_buffer));
        }
        sprintf(display_buffer,(char *)term_sequence);
        ret_code = write(1,display_buffer,strlen(display_buffer));
        wait_for_send();
}


put_stocklevel()
{
        int ret_code;

        sprintf(display_buffer,(char *)stock_output_screen,
                stocklevel.threshold,
                stocklevel.low_stock);
        ret_code = write(1,display_buffer,strlen(display_buffer));
        sprintf(display_buffer,(char *)term_sequence);
        ret_code = write(1,display_buffer,strlen(display_buffer));
        wait_for_send();
}

put_delivery()
{
        int ret_code;


        sprintf(display_buffer,(char *)delivery_output_screen,
                        delivery.o_carrier_id,
                        delivery.status);
        ret_code = write(1,display_buffer,strlen(display_buffer));
        sprintf(display_buffer,(char *)term_sequence);
        ret_code = write(1,display_buffer,strlen(display_buffer));
        wait_for_send();
}


cleanup(code)
{
        int ret_code;

        ret_code = write(1,cleanup_template,strlen(cleanup_template));
        if (code == TXERRCODE) {
                put_err(TXERRMSG);
                wait_for_send();
                return;
        }
        restore_tx();

        exit(code);
}


/*
 * File: del.ec
 * Delivery client Tuxedo code
 * Author : Shanti S
 * Date   : 8/04/93
 *
 */

#include <stdio.h>
#include <syslog.h>

#include "tpcc_client.h"

/* Tuxedo */
#include "atmi.h"
#include "Uunix.h"

struct req_struct *delp;
extern struct del_struct delivery;
/*
 * Function: init deltat
 * Init. Tuxedo
 */
init_del_tx()
{
        if ((delp = (struct req_struct *)tpalloc("CARRAY", NULL, sizeof(struct
req_struct))) == NULL) {
                syslog(LOG_ERR, "del tpalloc failed. tperrno = %d", tperrno);
                tpterm();
                cleanup(1);
        }
}


int
queue_delivery()
{
        ret_code = write(1,display_buffer,strlen(display_buffer));

        /* Init all fields to NULL */
        clr_fields(delivery_fld);
error_input:
        get_input(cur_field);
        cur_field = 0;

        if(get_integer(cp->value, cp->len, cur_field) !=0)
                goto error_input;
        if (*((int *)(cp->value)) == 0) {
                print_error_on_field();
                goto error_input;
        }
        return(0);
}


/* DISPLAY FUNCTIONS */
/* Display error message on last line of screen */
put_err(errmsg)
char *errmsg;
{
        sprintf(display_buffer,(char *)error_string,errmsg);
        write(1,display_buffer,strlen(display_buffer));
        err_flag = 1;
}

/* DISPLAY FUNCTIONS */
/* Display error message on last line of screen */
clear_err()
{
        sprintf(display_buffer,(char *)clear_error_string);
        write(1,display_buffer,strlen(display_buffer));
}



put_neworder(count)
int count;/* Count of items */
{
        struct fld *nextp = neworder_fld;
        struct no_itm_struct *valp = neworder.n_items;
        int i, cur_y;
        int ret_code;


        sprintf(display_buffer,(char *)clear_exec_string);
        write(1,display_buffer,strlen(display_buffer));
        sprintf(display_buffer,(char *)neworder_header_output_screen,
                        neworder.d_id,          /* d-id */
                        neworder.o_entry_d,        /* entry date */
                        neworder.c_id,          /* c-id */
                        neworder.c_last,
                        neworder.c_credit,          /* customer credit */
                        neworder.c_discount,       /* c-discount */
                        neworder.o_id,        /* order-id */
                        neworder.o_ol_cnt,    /* order-count */
                        neworder.w_tax,    /* w_tax */
                        neworder.d_tax,    /* d_tax */
                        neworder.status,          /* Execution status */
                        neworder.total);  /* Total amount */
        ret_code = write(1,display_buffer,strlen(display_buffer));
        for (i = 0; i < count; i++, valp++) {
        sprintf(display_buffer,(char *)neworder_line_output_screen,
                valp->ol_supply_w_id,
                valp->ol_i_id,
                valp->i_name,
                valp->ol_quantity,
                valp->s_quantity,
                valp->brand,
                valp->i_price,
                valp->ol_amount);
        ret_code = write(1,display_buffer,strlen(display_buffer));
        }
        for (i = count; i < 15; i++) {
        sprintf(display_buffer,(char *)neworder_blank_output_screen);
        ret_code = write(1,display_buffer,strlen(display_buffer));
        }
        sprintf(display_buffer,(char *)term_sequence);
        ret_code = write(1,display_buffer,strlen(display_buffer));
        wait_for_send();


}


put_payment()
{
int ret_code;

        sprintf(display_buffer,(char *)payment_output_screen,
                payment.h_date,
                payment.d_id,          /* d-id */
                payment.w_street_1,
                payment.d_street_1,
                payment.w_street_2,
                payment.d_street_2,
                payment.w_city,
                payment.w_state,
                payment.w_zip,
                payment.d_city,
                payment.d_state,
                payment.d_zip,
                payment.c_id,  /* c-id */
                payment.c_w_id,
                payment.c_d_id,
                payment.c_first,
                payment.c_middle,
                payment.c_last,  /* customer name */
                payment.c_since,
                payment.c_street_1,
                payment.c_credit,
                payment.c_street_2,
                payment.c_discount,
                payment.c_city,
                payment.c_state,
                payment.c_zip,
                payment.c_phone,
                payment.h_amount,
                payment.c_balance,
                payment.c_credit_lim,
                payment.c_data_1,
                payment.c_data_2,
                payment.c_data_3,
		strcpy(errmsg, "All required fields have not been entered
");
	else if (payment.c_id == 0 && payment.c_last[0] == '\0')
{
		strcpy(errmsg, "You must enter either the Customer-id or
Name");
}
	else if (payment.h_amount > 9999.99)
		strcpy(errmsg, "Invalid amount entered                ");
	else {
		if (err_flag) {
			err_flag = 0;
		clear_err();
		}
		return(0);
	}
	put_err(errmsg);
	goto error_input;
}


/*
 * Function: get_ordstat
 * Display ordstat form and get user input
 */


get_ordstat()
{
	int i, term = 0;/* Terminating character */
	struct fld *nextp;
	int ret_code;
	int cur_field,status;

	cur_field = 0;

	sprintf(display_buffer,(char *)orderstat_input_screen,w_id);
	ret_code = write(1,display_buffer,strlen(display_buffer));

	/* Init all fields to NULL */
	clr_fields(ordstat_fld);
error_input:
	get_input(cur_field);
	cur_field = 0;

	if(get_integer(&ordstat.d_id, 2, cur_field) != 0)
		goto error_input;
	cur_field++;
	if(get_integer(&ordstat.c_id, 4, cur_field) != 0)
		goto error_input;
	cur_field++;
	if(get_string(ordstat.c_last, 17, cur_field) != 0)
		goto error_input;
	cur_field++;

	/* Check if reqd. fields have been entered */
	if (ordstat.d_id == 0 ) {
		strcpy(errmsg, "All required fields have not been entered
");
	}
	else if (ordstat.c_id == 0 && ordstat.c_last[0] == '\0') {
		strcpy(errmsg, "You must enter either the Customer-id or
Name");
	}
	else {
		if (err_flag) {
			err_flag = 0;
			clear_err();
		}
		return(0);
	}
	put_err(errmsg);
	goto error_input;
}


/*
 * Function: get_stocklevel
 * Display stocklevel form and read user input
 */
get_stocklevel()
{
	struct fld *cp = &stocklevel_fld[0];/* threshold */
	int term;
	int ret_code;
	int cur_field,status;

	cur_field = 0;

	sprintf(display_buffer,(char *)stock_input_screen,w_id,d_id);
	ret_code = write(1,display_buffer,strlen(display_buffer));

	/* Init all fields to NULL */
	clr_fields(stocklevel_fld);
error_input:
	get_input(cur_field);
	cur_field = 0;


	/* Get threshold */
	if(get_integer(&stocklevel.threshold, 2, cur_field) !=0)
		goto error_input;

	if (*((int *)(cp->value)) == 0) {
	print_error_on_field();
		goto error_input;
	}
	return(0);
}


/*
 * Function: get_delivery
 * Display delivery form and read user input
 */
get_delivery()
{
	struct fld *cp = &delivery_fld[0];
	int term;
	int ret_code;
	int cur_field,status;

	cur_field = 0;

	sprintf(display_buffer,(char *)delivery_input_screen,w_id);
```

```
get_neworder()
{
        int num_lines, cur_y, i = 0, term = 0;/* Terminating character */
        struct fld *nextp;
        struct no_itm_struct *beginp, *curp;
        int allnulls;
        int ret_code;
        int cur_field,status;

        cur_field = 0;

        sprintf(display_buffer,(char *)neworder_input_screen,w_id);
        ret_code = write(1,display_buffer,strlen(display_buffer));
        /* Init. all fields to NULL */
        clr_fields(neworder_fld);
        for (curp = &neworder.n_items[0];
                        curp < &neworder.n_items[15]; curp++) {
                curp->ol_supply_w_id = curp->ol_i_id = curp->ol_quantity = 0;
        }

error_input:
        get_input(cur_field);
        cur_field = 0;
        if(get_integer(&neworder.d_id, 2, cur_field) != 0)
                goto error_input;
        cur_field++;
        if(get_integer(&neworder.c_id, 4, cur_field) != 0)
                goto error_input;
        cur_field++;
        beginp = &neworder.n_items[0];
        curp = beginp;
        for (i=0 ; i < 15; i++) {
                if(get_integer(&curp->ol_supply_w_id, 4, cur_field) != 0)
                        goto error_input;
                cur_field++;
                if(get_integer(&curp->ol_i_id, 6, cur_field) != 0)
                        goto error_input;
                cur_field++;
                if(get_integer(&curp->ol_quantity, 2, cur_field) != 0)
                        goto error_input;
                cur_field++;
                curp++;
        }


        if (neworder.d_id == 0 || neworder.c_id == 0) {
                        strcpy(errmsg, "All required fields have not been
entered");
                        put_err(errmsg);
                        goto error_input;
        }
        /* Check if all fields entered for each line item */
        allnulls = 0;
        num_lines = 15;
        for (i = 0; i < 15; i++) {
                if (neworder.n_items[i].ol_supply_w_id == 0) {
                        if (neworder.n_items[i].ol_i_id == 0 &&
                                neworder.n_items[i].ol_quantity == 0) {
                                if ( !allnulls) {/* First null line */
                                        num_lines = i;/* Number of line items */
                                        allnulls = 1;
                                }
                                continue;
                        }
                }
                else if (allnulls == 1)
                        goto error;
                else if (neworder.n_items[i].ol_i_id != 0 &&
                                neworder.n_items[i].ol_quantity != 0)
                        continue;
error:
        strcpy(errmsg, "Invalid item-line entered                   ");
        put_err(errmsg);
        goto error_input;
        }

        if (err_flag) {
                err_flag = 0;
                clear_err();
        }
        return(num_lines);
}

/*
 * Function: get_payment
 * Display payment form and get user input
 */
get_payment()
{
        int i, term = 0;/* Terminating character */
        struct fld *nextp;
        int ret_code;
        int cur_field,status;

        cur_field = 0;

        sprintf(display_buffer,(char *)payment_input_screen,w_id);
        ret_code = write(1,display_buffer,strlen(display_buffer));

        /* Init all fields to NULL */
        clr_fields(payment_fld);
error_input:
        get_input(cur_field);
        cur_field = 0;

        if(get_integer(&payment.d_id, 2, cur_field) != 0)
                goto error_input;
        cur_field++;
        if(get_integer(&payment.c_id, 4, cur_field) != 0)
                goto error_input;
        cur_field++;
        if(get_integer(&payment.c_w_id, 4, cur_field) != 0)
                goto error_input;
        cur_field++;
        if(get_integer(&payment.c_d_id, 2, cur_field) != 0)
                goto error_input;
        cur_field++;
        if(get_string(payment.c_last, 17, cur_field) != 0)
                goto error_input;
        cur_field++;
        if(get_amount(&payment.h_amount, 8, cur_field) != 0)
                goto error_input;
        cur_field++;
                        /* Check if reqd. fields have been entered */
        if (payment.d_id == 0 || payment.c_d_id == 0 ||
                payment.c_w_id == 0 || payment.h_amount == 0.0)
```

```
27,’-’,’0’,’5’,’\ \’,’T’,’o’,’t’,’a’,’l’,’:’,
27,’&’,              /* Protect mode */
27,’N’,              /* Page edit mode */
27,’(‘,             /* unprotect */
27,’0’,’1’,’S’,      /* Program send key */
27,’B’, 0 };         /* Block Mode */

unsigned char error_string[] = {
27,0x27,27,’)’,27,’-’,’0’,’6’,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’-’,’0’,’6’,’
‘,’%’,’s’,27,’p’,’@’,
27,’&’,               /* Protect mode */
27,’N’,              /* Page edit mode */
27,’(‘,              /* unprotect */
27,’0’,’1’,’S’,      /* Program send key */
27,’B’,  0 };         /* Block Mode */


unsigned char clear_exec_string[] = {
27,0x27,27,’)’,27,’-’,’0’,’5’,
‘ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,
‘ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,
‘ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,
‘ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,
27,’p’,’@’,
27,’&’,               /* Protect mode */
27,’N’,              /* Page edit mode */
27,’(‘,              /* unprotect */
27,’0’,’1’,’S’,      /* Program send key */
27,’B’,0};

unsigned char clear_error_string[] = {
27,0x27,27,’)’,27,’-’,’0’,’6’,
‘ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,
‘ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,
‘ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,
‘ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,
27,’p’,’@’,
27,’&’,               /* Protect mode */
27,’N’,              /* Page edit mode */
27,’(‘,              /* unprotect */
27,’0’,’1’,’S’,      /* Program send key */
27,’B’,0};



 /*
  * Function: put_menu
  * Displays the menu and gets user’s selection
  */
put_menu()
{
        char choice;
        int  result = 0;
        int ret_code;

        ret_code = write(1,menu_screen,strlen(menu_screen));
        while( ! result) {
                get_input(0);
                get_char(&choice ,0); /* Get a single character response */
                switch(choice) {
                case ‘n’:
                case ‘N’:
                        result = (NEWORDER);
                        break;
                case ‘p’:
                case ‘P’:
                        result = (PAYMENT);
                        break;
                case ‘o’:
                case ‘O’:
                        result = (ORDSTAT);
                        break;
                case ‘d’:
                case ‘D’:
                        result = (DELIVERY);
                        break;
                case ‘s’:
                case ‘S’:
                        result = (STOCKLEVEL);
                        break;
                case ‘e’:
                case ‘E’:
                        result = (EXIT);
                        break;
                default:
                        print_error_on_field();
                }
        }
        return(result);
}


 /*
  * Function: clr_fields
  * This function clears the field values of all the
  * fields in the given fld structure
  */
clr_fields(fldp)
struct fld *fldp;
{
        struct fld *nextp = fldp;

        while ( nextp->type != -1 ) {
                if (nextp->input_allowed) {
                switch (nextp->type) {
                case INT:
                        *((int *)(nextp->value)) = 0;
                        break;
                case AMOUNT:
                        *((double *)(nextp->value)) = 0.0;
                        break;
                case STRING:
                        *(nextp->value) = 0;
                        break;
                }
                }
                nextp++;
        }
}


 /*
  * Function: get_neworder
  * Display neworder form and get user input
  * Return # of items entered
  */
```

```
0x1f,0x1f,’C’,’u’,’s’,’t’,’-’,’D’,’a’,’t’,’a’,’:’,’ ‘,
27,’&’,              /* Protect mode */
27,’N’,              /* Page edit mode */
27,’(‘,             /* unprotect */
27,’0’,’1’,’S’,      /* Program send key */
27,’B’, 0 };         /* Block Mode */



unsigned char orderstat_input_screen[] = {
27,’*’,      /* Clear to nulls */
27,’,’,      /* Clear to protected */
27,0x22,     /* Input to numeric */
27,’)’,27,’G’,’8’,       /* Set underline mode */
27,’-’,’0’,’ ‘,’B’,’O’,’r’,’d’,’e’,’r’,’-’,’S’,’t’,’a’,’t’,’u’,’s’,
0x1f,’W’,’a’,’r’,’e’,’h’,’o’,’u’,’s’,’e’,’:’,’ ‘,’%’,’0’,’4’,’d’,’ ‘,’ ‘,
‘D’,’i’,’s’,’t’,’r’,’i’,’c’,’t’,’:’,’ ‘,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’C’,’u’,’s’,’t’,’o’,’m’,’e’,’r’,’:’,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,
‘ ‘,’ ‘, ‘N’,’a’,’m’,’e’,’:’,’ ‘,
27,’-’,’0’,0x22,’J’,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,
0x1f,’C’,’u’,’s’,’t’,’-’,’B’,’a’,’l’,’a’,’n’,’c’,’e’,’:’,’ ‘,
0x1f,0x1f,’O’,’r’,’d’,’e’,’r’,’-’,’N’,’u’,’m’,’b’,’e’,’r’,’:’,’ ‘,
27,’-’,’0’,0x25,0x25,’8’,’E’,’n’,’t’,’r’,’y’,’-’,’D’,’a’,’t’,’e’,’:’,
27,’-’,’0’,0x25,0x25,’Z’,’C’,’a’,’r’,’r’,’i’,’e’,’r’,’-’,’N’,’u’,’m’,’b’,’e’,’r’,’:’,’ ‘,
0x1f,’S’,’u’,’p’,’p’,’_’,’W’,’ ‘,’ ‘,’I’,’t’,’e’,’m’,’_’,’I’,’d’,’ ‘,’ ‘,
‘Q’,’t’,’y’,’ ‘,’ ‘,’A’,’m’,’o’,’u’,’n’,’t’,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’D’,’e’,’l’,’i’,’v’,’e’,’r’,’y’,’-
’,’D’,’a’,’t’,’e’,
27,’&’,              /* Protect mode */
27,’N’,              /* Page edit mode */
27,’(‘,             /* unprotect */
27,’0’,’1’,’S’,      /* Program send key */
27,’B’, 0 };         /* Block Mode */



unsigned char delivery_input_screen[] = {
27,’*’,      /* Clear to nulls */
27,’,’,      /* Clear to protected */
27,0x22,     /* Input to numeric */
27,’)’,27,’G’,’8’,       /* Set underline mode */
27,’-’,’0’,’ ‘,’B’,’D’,’e’,’l’,’i’,’v’,’e’,’r’,’y’,
0x1f,’W’,’a’,’r’,’e’,’h’,’o’,’u’,’s’,’e’,’:’,’ ‘,’%’,’0’,’4’,’d’,’ ‘,’ ‘,’ ‘,
0x1f,0x1f,’C’,’a’,’r’,’r’,’i’,’e’,’r’,’ ‘,’N’,’u’,’m’,’b’,’e’,’r’,’:’,’ ‘,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,0x1f,’E’,’x’,’e’,’c’,’u’,’t’,’i’,’o’,’n’,’ ‘,’S’,’t’,’a’,’t’,’u’,’s’,’:’,’ ‘,
27,’&’,              /* Protect mode */
27,’N’,              /* Page edit mode */
27,’(‘,             /* unprotect */
27,’0’,’1’,’S’,      /* Program send key */
27,’B’, 0 };         /* Block Mode */



unsigned char stock_input_screen[] = {
27,’*’,      /* Clear to nulls */
27,’,’,      /* Clear to protected */
27,0x22,     /* Input to numeric */
27,’)’,27,’G’,’8’,       /* Set underline mode */
27,’-’,’0’,’ ‘,’B’,’S’,’t’,’o’,’c’,’k’,’-’,’l’,’e’,’v’,’e’,’l’,
0x1f,’W’,’a’,’r’,’e’,’h’,’o’,’u’,’s’,’e’,’:’,’ ‘,’%’,’0’,’4’,’d’,’ ‘,’ ‘,’ ‘,
‘ ‘,’ ‘,’ ‘,’D’,’i’,’s’,’t’,’r’,’i’,’c’,’t’,’:’,’ ‘,’%’,’0’,’2’,’d’,’ ‘,’ ‘,’ ‘,
0x1f,0x1f,’S’,’t’,’o’,’c’,’k’,’ ‘,’L’,’e’,’v’,’e’,’l’,
‘ ‘,’T’,’h’,’r’,’e’,’s’,’h’,’o’,’l’,’d’,’:’,’ ‘,
27,’(‘,’ ‘,’ ‘,27,’)’,
```

```
0x1f,0x1f,’L’,’o’,’w’,’ ‘,’S’,’t’,’o’,’c’,’k’,’:’,’ ‘,
27,’&’,              /* Protect mode */
27,’N’,              /* Page edit mode */
27,’(‘,             /* unprotect */
27,’0’,’1’,’S’,      /* Program send key */
27,’B’, 0 };         /* Block Mode */


unsigned char neworder_input_screen[] = {
27,’*’,      /* Clear to nulls */
27,’,’,      /* Clear to protected */
27,0x22,     /* Input to numeric */
27,’)’,27,’G’,’8’,       /* Set underline mode */
27,’-’,’0’,’ ‘,’B’,’N’,’e’,’w’,’ ‘,’O’,’r’,’d’,’e’,’r’,
0x1f,’W’,’a’,’r’,’e’,’h’,’o’,’u’,’s’,’e’,’:’,’ ‘,’%’,’0’,’4’,’d’,’ ‘,’ ‘,’ ‘,
‘D’,’i’,’s’,’t’,’r’,’i’,’c’,’t’,’:’,’ ‘,27,’(‘,’ ‘,’ ‘,27,’)’,
27,’-’,’0’,’!’,’U’,’D’,’a’,’t’,’e’,’:’,
0x1f,’C’,’u’,’s’,’t’,’o’,’m’,’e’,’r’,’:’,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,
‘ ‘,’ ‘,’ ‘,’N’,’a’,’m’,’e’,’:’,’ ‘,
27,’-’,’0’,0x22,’J’,’C’,’r’,’e’,’d’,’i’,’t’,’:’,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,
‘%’,’D’,’i’,’s’,’c’,’:’,’ ‘,
0x1f,’O’,’r’,’d’,’e’,’r’,’ ‘,’N’,’u’,’m’,’b’,’e’,’r’,’:’,
‘ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,
‘N’,’u’,’m’,’b’,’e’,’r’,’ ‘,’o’,’f’,’ ‘,’L’,’i’,’n’,’e’,’s’,’:’,
‘ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,
‘W’,’_’,’t’,’a’,’x’,’:’,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,
‘D’,’_’,’t’,’a’,’x’,’:’,
0x1f,0x1f,’S’,’u’,’p’,’p’,’_’,’W’,
‘ ‘,’ ‘,’I’,’t’,’e’,’m’,’_’,’I’,’d’,’ ‘,’ ‘,
‘I’,’t’,’e’,’m’,’ ‘,’N’,’a’,’m’,’e’,
27,’]’,’K’,’Q’,’t’,’y’,’ ‘,’ ‘,’S’,’t’,’o’,’c’,’k’,’ ‘,’ ‘,’B’,’/’,’G’,
‘ ‘,’ ‘,’P’,’r’,’i’,’c’,’e’,’ ‘,’ ‘,’ ‘,’ ‘,’A’,’m’,’o’,’u’,’n’,’t’,
0x1f,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,
27,’]’,’K’,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,
27,’]’,’K’,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,
27,’]’,’K’,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,
27,’]’,’K’,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,
27,’]’,’K’,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,
27,’]’,’K’,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,
27,’]’,’K’,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,
27,’]’,’K’,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,
27,’]’,’K’,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,
27,’]’,’K’,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,
27,’]’,’K’,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,
27,’]’,’K’,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,
27,’]’,’K’,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,
27,’]’,’K’,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,
27,’]’,’K’,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’E’,’x’,’e’,’c’,’u’,’t’,’i’,’o’,’n’,’ ‘,’S’,’t’,’a’,’t’,’u’,’s’,’:’,
27,’]’,’X’,27,’)’,’%’,’3’,’s’,                     /* payment.c_credit */
0x1f,27,’]’,0x27,27,’)’,’%’,’s’,                      /* payment.c_street_2 */
27,’]’,’X’,27,’)’,’%’,’0’,’5’,’.’,’2’,’f’,         /* payment.c_discount */
0x1f,27,’]’,0x27,27,’)’,’%’,’s’,                      /* payment.c_city */
27,’]’,’<‘,27,’)’,’%’,’s’,                        /* payment.c_state */
27,’]’,’?’,27,’)’,’%’,’s’,                        /* payment.c_zip */
27,’]’,’X’,27,’)’,’%’,’s’,                        /* payment.c_phone */
0x1f,0x1f,27,’]’,’5’,27,’)’,’$’,’%’,’0’,’7’,’.’,’2’,’f’,   /* payment.h_amount */
27,’]’,’U’,27,’)’,’$’,’%’,’0’,’1’,’4’,’.’,’2’,’f’,        /* payment.c_balance */
0x1f,27,’]’,’ / ’,27,’)’,’$’,’%’,’0’,’1’,’3’,’.’,’2’,’f’,/* payment.c_credit_lim*/
0x1f,0x1f,27,’]’,’*’,27,’)’,’%’,’s’,              /* payment.c_data_1 */
0x1f,27,’]’,’*’,27,’)’,’%’,’s’,                  /* payment.c_data_2 */
0x1f,27,’]’,’*’,27,’)’,’%’,’s’,                  /* payment.c_data_3 */
0x1f,27,’]’,’*’,27,’)’,’%’,’s’,                  /* payment.c_data_4 */
0 };


/* This sequence is output at the end of each screen so the slave can
detect */
/* end of output                                                      */

unsigned char term_sequence[] = {
27,’-’,’0’,’6’,’j’,27,’)’,’*’,’*’,’(‘,’(‘,0};       /* TERM_SEQ  */



unsigned char order_line_output_screen[] = {
0x1f,27,’]’,’!’,’%’,’0’,’4’,’d’,                  /* ol_supply_w_id */
27,’]’,’(‘,’%’,’0’,’6’,’d’,                    /* ol_i_id */
27,’]’,’1’,’%’,’0’,’2’,’d’,                     /* ol_quantity */
27,’]’,’6’,’%’,’0’,’9’,’.’,’0’,’2’,’f’,           /* ol_amount */
27,’]’,’D’,’%’,’s’,                            /* ol_delivery_d */
0};



unsigned char ordstat_header_output_screen[] = {
27,’;’,27,’+’,0x1a,27,0x27,27,’O’,27,’G’,’0’,
27,’-’,’0’,’!’,’;’,27,’)’,’%’,’0’,’2’,’d’,                /* ordstat.d_id */
0x1f,27,’]’,’*’,27,’)’,’%’,’0’,’4’,’d’,                    /* ordstat.c_id */
27,’]’,’6’,27,’)’,’%’,’1’,’7’,’s’,                      /* ordstat.c_first */
27,’]’,’G’,27,’)’,’%’,’3’,’s’,                          /* ordstat.c_middle */
27,’]’,’J’,27,’)’,’%’,’1’,’7’,’s’,                     /* ordstat.c_last */
0x1f,27,’]’,’-’,27,’)’,’$’,’%’,’0’,’9’,’.’,’0’,’2’,’f’,   /* ordstat.c_balance */
0x1f,0x1f,27,’]’,’-’,27,’)’,’%’,’0’,’8’,’d’,              /* ordstat.o_id */
27,’]’,’E’,27,’)’,’%’,’s’,                             /* ordstat.o_entry_d */
27,’]’,’j’,27,’)’,’%’,’0’,’2’,’d’,                  /* ordstat.o_carrier_id */
27,’-’,’0’,’&’,’&’,0x1f,       /* one before lines because cr at start of ol */
0 };



unsigned char stock_output_screen[] = {
27,’;’,27,’+’,0x1a,27,0x27,27,’O’,27,’G’,’0’,
27,’-’,’0’,’#’,’7’,27,’)’,’%’,’0’,’2’,’d’,          /* stocklevel.threshold */
0x1f,0x1f,27,’]’,’*’,27,’)’,’%’,’0’,’3’,’d’,         /* stocklevel.low_stock */
0 };



unsigned char delivery_output_screen[] = {
27,’;’,27,’+’,0x1a,27,0x27,27,’O’,27,’G’,’0’,
27,’-’,’0’,’#’,’0’,27,’)’,’%’,’0’,’2’,’d’,          /* delivery.o_carrier_id */
0x1f,0x1f,27,’]’,’2’,27,’)’,’%’,’s’,                 /* delivery.status */
0 };

unsigned char neworder_header_output_screen[] = {
27,’;’,27,’+’,0x1a,27,0x27,27,’O’,27,’G’,’0’,
27,’-’,’0’,’!’,’<‘,27,’)’,’%’,’0’,’2’,’d’,          /* neworder.d_id */
27,’]’,’[‘,27,’)’,’%’,’s’,                      /* neworder.o_entry_d */
27,’-’,’0’,0x22,’*’,27,’)’,’%’,’0’,’4’,’d’,          /* neworder.c_id */
27,’]’,’7’,27,’)’,’%’,’s’,                      /* neworder.c_last */
27,’]’,’R’,27,’)’,’%’,’s’,                      /* neworder.c_credit */
27,’]’,’^’,27,’)’,’%’,’0’,’5’,’.’,’0’,’2’,’f’,       /* neworder.c_discount */
27,’-’,’0’,’#’,’-’,27,’)’,’%’,’0’,’8’,’d’,          /* neworder.o_id */
27,’]’,’H’,27,’)’,’%’,’0’,’2’,’d’,                 /* neworder.o_ol_cnt */
27,’]’,’Y’,27,’)’,’%’,’0’,’5’,’.’,’0’,’2’,’f’,       /* neworder.w_tax */
27,’]’,’h’,27,’)’,’%’,’0’,’5’,’.’,’0’,’2’,’f’,       /* neworder.d_tax */
27,’-’,’0’,’5’,’ ‘,’E’,’x’,’e’,’c’,’u’,’t’,’i’,’o’,’n’,’
‘,’S’,’t’,’a’,’t’,’u’,’s’,’:’,27,’)’,’%’,’s’,                /* neworder.status */
27,’]’,’d’,27,’)’,’$’,’%’,’0’,’8’,’.’,’0’,’2’,’f’,  /* neworder.total */
27,’-’,’0’,’%’,’%’,0x1f,       /* one before lines because cr at start of ol */
0 };


/* For each unused line of neworder output send this sequence */
unsigned char neworder_blank_output_screen[] = { 0x1f,27,’T’,27,’)’,0};


unsigned char neworder_line_output_screen[] = {
0x1f,27,’]’,’!’,’%’,’0’,’4’,’d’,                  /* ol_supply_w_id */
27,’]’,’(‘,’%’,’0’,’6’,’d’,                    /* ol_i_id */
27,’]’,’1’,’%’,’s’,                          /* i_name */
27,’]’,’K’,’%’,’0’,’2’,’d’,                     /* ol_quantity */
27,’]’,’Q’,’%’,’0’,’3’,’d’,                     /* s_quantity */
27,’]’,’X’,’%’,’s’,                           /* brand */
27,’]’,’ \ \ ’,’$’,’%’,’0’,’6’,’.’,’0’,’2’,’f’,      /* i_price */
27,’]’,’e’,’$’,’%’,’0’,’7’,’.’,’0’,’2’,’f’,        /* ol_amount */
0};


/* The input screens.  The input fields are delimited by 27,( sp .. sp 27,) */
/* This is set unprotect/set protect.  The user will only be able to enter  */
/* data on the spaces.  The rest is just the text output                    */

unsigned char payment_input_screen[] = {
27,’*’,      /* Clear to nulls */
27,’,’,      /* Clear to protected */
27,0x22,     /* Input to numeric */
27,’)’,27,’G’,’8’,       /* Set underline mode */
27,’-’,’0’,’ ‘,’B’,’P’,’a’,’y’,’m’,’e’,’n’,’t’,
0x1f,’D’,’a’,’t’,’e’,’:’,
0x1f,0x1f,’W’,’a’,’r’,’e’,’h’,’o’,’u’,’s’,’e’,’:’,’ ‘,’%’,’0’,’4’,’d’,
27,’]’,’G’,’D’,’i’,’s’,’t’,’r’,’i’,’c’,’t’,’:’,’ ‘,27,’(‘,’ ‘,’ ‘,27,’)’,
27,’-’,’0’,’(‘,’ ‘,’C’,’u’,’s’,’t’,’o’,’m’,’e’,’r’,’:’,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,’ ‘,
‘C’,’u’,’s’,’t’,’-’,’W’,’a’,’r’,’e’,’h’,’o’,’u’,’s’,’e’,’:’,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,’ ‘,
‘C’,’u’,’s’,’t’,’-’,’D’,’i’,’s’,’t’,’r’,’i’,’c’,’t’,’:’,’ ‘,27,’(‘,’ ‘,’ ‘,27,’)’,
0x1f,’N’,’a’,’m’,’e’,’:’,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,
27,’-’,’0’,’)’,’P’,’S’,’i’,’n’,’c’,’e’,’:’,’ ‘,
27,’-’,’0’,’*’,’P’,’C’,’r’,’e’,’d’,’i’,’t’,’:’,’ ‘,
27,’-’,’0’,’+’,’P’,’%’,’D’,’i’,’s’,’c’,’:’,’ ‘,
27,’-’,’0’,0x2c,’P’,’P’,’h’,’o’,’n’,’e’,’:’,
0x1f,0x1f,’A’,’m’,’o’,’u’,’n’,’t’,’ ‘,’P’,’a’,’i’,’d’,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’(‘,’ ‘,’ ‘,’ ‘,’
‘,’ ‘,’ ‘,’ ‘,’ ‘,27,’)’,
27,’-’,’0’,0x2e,’C’,’N’,’e’,’w’,’ ‘,’C’,’u’,’s’,’t’,’-’,’B’,’a’,’l’,’a’,’n’,’c’,’e’,’:’,’ ‘,
0x1f,’C’,’r’,’e’,’d’,’i’,’t’,’ ‘,’L’,’i’,’m’,’i’,’t’,’:’,’ ‘,
struct fld payment_fld[] = {
INT, 2, 51, 4, 1, (char *)&payment.d_id,  /* d-id */
INT, 4, 10, 9, 1, (char *)&payment.c_id,  /* c-id */
INT, 4, 32, 9, 1, (char *)&payment.c_w_id,
INT, 2, 53, 9, 1, (char *)&payment.c_d_id,
STRING, 17, 28, 10, 1, payment.c_last,  /* customer name */
AMOUNT, 8, 22, 15, 1, (char *)&payment.h_amount,
STRING, 20, 0, 2, 0, payment.h_date,
INT, 4, 11, 4, 0, (char *)&payment.w_id,  /* w-id */
STRING, 21, 0, 5, 0, payment.w_street_1,
STRING, 21, 0, 6, 0, payment.w_street_2,
STRING, 21, 0, 7, 0, payment.w_city,
STRING, 3, 21, 7, 0, payment.w_state,
STRING, 11, 24, 7, 0, payment.w_zip,
STRING, 21, 41, 5, 0, payment.d_street_1,
STRING, 21, 41, 6, 0, payment.d_street_2,
STRING, 21, 41, 7, 0, payment.d_city,
STRING, 3, 62, 7, 0, payment.d_state,
STRING, 11, 65, 7, 0, payment.d_zip,
STRING, 17, 8, 10, 0, payment.c_first,
STRING, 3, 25, 10, 0, payment.c_middle,
STRING, 21, 8, 11, 0, payment.c_street_1,
STRING, 21, 8, 12, 0, payment.c_street_2,
STRING, 21, 8, 13, 0, payment.c_city,
STRING, 3, 29, 13, 0, payment.c_state,
STRING, 11, 32, 13, 0, payment.c_zip,
STRING, 11, 57, 10, 0, payment.c_since,
STRING, 3, 57, 11, 0, payment.c_credit,
AMOUNT, 15, 54, 15, 0, (char *)&payment.c_balance,
AMOUNT, 14, 16, 16, 0, (char *)&payment.c_credit_lim,
AMOUNT, 5, 57, 12, 0, (char *)&payment.c_discount,
STRING, 51, 11, 18, 0, payment.c_data_1,
STRING, 51, 11, 19, 0, payment.c_data_2,
STRING, 51, 11, 20, 0, payment.c_data_3,
STRING, 51, 11, 21, 0, payment.c_data_4,
STRING, 20, 57, 13, 0, payment.c_phone,
-1, -1, -1, -1, -1, (char *)NULL
};


struct fld delivery_fld[] = {
INT, 2, 16, 4, 1, (char *)&delivery.o_carrier_id,
INT, 4, 11, 2, 0, (char *)&delivery.w_id,
STRING, 26, 18, 6, 0, delivery.status,
-1, -1, -1, -1, -1, (char *)NULL
};


struct fld stocklevel_fld[] = {
INT, 2, 23, 4, 1, (char *)&stocklevel.threshold,
INT, 4, 11, 2, 0, (char *)&stocklevel.w_id,
INT, 2, 28, 2, 0, (char *)&stocklevel.d_id,
INT, 3, 11, 6, 0, (char *)&stocklevel.low_stock,
-1, -1, -1, -1, -1, (char *)NULL
};


struct fld ordstat_fld[] = {
INT, 2, 28, 2, 1, (char *)&ordstat.d_id,
INT, 4, 10, 3, 1, (char *)&ordstat.c_id,
STRING, 17, 43, 3, 1, ordstat.c_last,
INT, 4, 11, 2, 0, (char *)&ordstat.w_id,
STRING, 17, 23, 3, 0, ordstat.c_first,  /* customer name */
STRING, 3, 40, 3, 0, ordstat.c_middle,
AMOUNT, 10, 14, 4, 0, (char *)&ordstat.c_balance,
INT, 8, 14, 6, 0, (char *)&ordstat.o_id,  /* order-id */
STRING, 20, 37, 6, 0, ordstat.o_entry_d,  /* entry date */
INT, 2, 75, 6, 0, (char *)&ordstat.o_carrier_id,
-1, -1, -1, -1, -1, (char *)NULL
};


/* Sent on termination to reset the terminal to a clean state */
unsigned char cleanup_template[] = {
27, 42, 27, 104, 27, 46, 50, 27, 45, 48, 55, 27, 67, 27, 68, 70, 27, 71, 48,
0 };

/* The menu screen.  Presents a single character input field */

unsigned char menu_screen[] = {
27,'*',      /* Clear to nulls */
27,',',      /* Clear to protected */
27,0x22,     /* Set input type to numeric */
27,')','N','e','w','-','O','r','d','e','r','(','n',')',
' ',' ','P','a','y','m','e','n','t','(','p',')',
' ',' ','O','r','d','e','r','-','S','t','a','t','u','s','(','o',')',
' ',' ','D','e','l','i','v','e','r','y','(','d',')',
' ',' ','S','t','o','c','k','-','L','e','v','e','l','(','s',')',
' ',' ','E','x','i','t','(','e',')',
27,'(',' ', 27,')',
27,'&',
27,'N',
27,'(',
27,'0','1','S',
27,'B', 0 };


/* The output screens  These are basically a set of sprintf strings for the */
/* various put functions.   The comment beside each line indicates the record
*/
/* entry loaded into each field                                           */

unsigned char payment_output_screen[] = {
27,';',27,'+',0x1a,27,0x27,27,'O',27,'G','0',
27,'-','0','!',' ',27,')','%','s',                 /* payment.h_date */
27,'-','0','#','Q',27,')','%','0','2','d',        /* payment.d_id */
0x1f,27,')','%','s',                               /* payment.w_street_1 */
27,']','H',27,')','%','s',                         /* payment.d_street_1 */
0x1f,27,')','%','s',                               /* payment.w_street_2 */
27,']','H',27,')','%','s',                         /* payment.d_street_2 */
0x1f,27,')','%','s',                               /* payment.w_city */
27,']','4',27,')','%','s',                         /* payment.w_state */
27,']','7',27,')','%','s',                         /* payment.w_zip */
27,']','H',27,')','%','s',                         /* payment.d_city */
27,']',']',27,')','%','s',                         /* payment.d_state */
27,']','`',27,')','%','s',                         /* payment.d_zip */
0x1f,0x1f,27,']','+',27,')','%','0','4','d',      /* payment.c_id */
27,']','A',27,')','%','0','4','d',                /* payment.c_w_id */
27,']','U',27,')','%','0','2','d',                /* payment.c_d_id */
0x1f,27,']','&',27,')','%','1','7','s',          /* payment.c_first */
27,']','8',27,')','%','3','s',                    /* payment.c_middle */
27,']',';',27,')','%','1','7','s',                /* payment.c_last */
27,']','X',27,')','%','s',                         /* payment.c_since */
0x1f,27,']',0x27,27,')','%','s',                    /* payment.c_street_1 */
```

```
init_ordstat()
{
        ordstat.w_id = w_id;
        init_ords_tx();
}

init_delivery()
{
        delivery.w_id = w_id;
        init_del_tx();
}

init_stocklevel()
{
        stocklevel.w_id = w_id;
        stocklevel.d_id = d_id;
        init_stock_tx();
}

do_neworder()
{
        int no_of_items;

        no_of_items = get_neworder();
        if (no_of_items == -1)
                return;
        if (neworder_tx(no_of_items))
                cleanup(TXERRCODE);
        else
{
                put_neworder(no_of_items);
}
}

do_payment()
{
        if (get_payment() == -1)
                return;
        if (payment_tx())
                cleanup(TXERRCODE);
        else
{
                put_payment();
}
}

do_delivery()
{
        if (get_delivery() == -1)
                return;
        if (queue_delivery())
                cleanup(TXERRCODE);
        else
                put_delivery();
}

do_stocklevel()
{
        if (get_stocklevel() == -1)
                return;
        if (stocklevel_tx())
                cleanup(TXERRCODE);
        else
                put_stocklevel();
}


do_ordstat()
{
        int no_of_items;

        if (get_ordstat() == -1)
                return;
        no_of_items = ordstat_tx();
        if (no_of_items == TXERRCODE)
                cleanup(TXERRCODE);
        else
                put_ordstat(no_of_items);
}


/*
 * (c) Copyright Sun Microsystems Inc. 1994
 * Written: 15/10/94
 *
 */


#include <syslog.h>
#include <errno.h>
#include <stdio.h>
#include "tpcc_term.h"
#include "tpcc_client.h"



char display_buffer[3000];
static char errmsg[80];         /* For error messages */
char blank_mesg[70]= "                                                                      ";
static int      print_dollar;
static int      err_flag;               /* Flag to indicate if errmsg printed */


extern int current_max_field;



/* Form field structures */
struct fld neworder_fld[16] = {
INT, 2, 28, 2, 1, (char *)&neworder.d_id, /* d-id */
INT, 4, 11, 3, 1, (char *)&neworder.c_id, /* c-id */
INT, 4, 11, 2, 0, (char *)&neworder.w_id, /* w-id */
STRING, 20, 60, 2, 0, neworder.o_entry_d, /* entry date */
STRING, 17, 24, 3, 0, neworder.c_last,
STRING, 3, 51, 3, 0, neworder.c_credit, /* customer credit */
AMOUNT, 5, 63, 3, 0, (char *)&neworder.c_discount, /* c-discount */
INT, 8, 14, 4, 0, (char *)&neworder.o_id, /* order-id */
INT, 2, 41, 4, 0, (char *)&neworder.o_ol_cnt, /* order-count */
AMOUNT, 5, 58, 4, 0, (char *)&neworder.w_tax, /* w_tax */
AMOUNT, 5, 73, 4, 0, (char *)&neworder.d_tax, /* d_tax */
STRING, 25, 18, 22, 0, neworder.status, /* Execution status */
AMOUNT, 9, 69, 22, 0, (char *)&neworder.total, /* Total amount */
-1, -1, -1, -1, -1, (char *)NULL
};
```

Appendix A: Application Code

This Appendix contains the application source code that implements the transactions and Forms modules.

```
/*
 * (c) Copyright Sun Microsystems Inc. 1994
 * Written: 15/10/94
 *
 */
#include <syslog.h>
#include <errno.h>
#include <stdio.h>
#include "tpcc_client.h"

int w_id = 1, d_id = 1, queue_no = 1;/* Global w-id, d_id */

/* Field structures */
struct no_struct neworder;
struct pay_struct payment;
struct del_struct delivery;
struct stock_struct stocklevel;
struct ord_struct ordstat;
char newo_service[5] = "NEWO", paym_service[5] = "PAYM";
char ords_service[5] = "ORDS", del_service[4] = "DEL";
char stock_service[6] = "STOCK";

main()
{

	do_init();

	while (1) {
	switch(put_menu()) {
	case NEWORDER:
		do_neworder();
		break;
	case PAYMENT:
		do_payment();
		break;
	case DELIVERY:
		do_delivery();
		break;
	case STOCKLEVEL:
		do_stocklevel();
		break;
	case ORDSTAT:
		do_ordstat();
		break;
	case EXIT:
		cleanup(0);
	}
	}
}


do_init()
{
	openlog("Client", LOG_PID|LOG_CONS, 0);
	init_scr();		/* Init screen */
	init_neworder();
	init_payment();
	init_ordstat();
	init_delivery();
	init_stocklevel();
}


init_neworder()
{
	neworder.w_id = w_id;/* Post wid of this client */
	init_newo_tx();
}

init_payment()
{
	payment.w_id = w_id;
	init_paym_tx();
}
```

8.4 Availability

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

The entire configuration will be available by June 30, 1996.

8.5 TpmC, Price/TpmC

A statement of the measured tpmC, as well as the respective calculations for 5-year pricing, price/performance (price/tpmC), and the availability date must be included.

The Maximum Qualified Throughput for the Ultra Enterprise 150 was 1332.50 tpmC at $189.46 per tpmC. The availability date for the entire configuration is June 30, 1996.

9 - Clause 8 Related Items

9.1 Auditor's Report

The auditor's name, address, phone number, and a copy of the auditor's attestation letter indicating compliance must be included in the Full Disclosure Report.

See attached attestation letter for the Sun report as well as the Auditor's name and address.

The Executive Summary on page vi lists pricing information for all components. All Sun pricing is from CAT Technology, Inc. The hub pricing was obtained from Data Comm Warehouse. Please refer to appendix G for the quotes.

8.2 Support Pricing

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

8.2.1 Sun Hardware and Software Support

The Silver Program of the SunService Support Program was used in all Sun pricing calculations. This program provides complete service with both on-site and telephone assistance. Features of this program include telephone assistance from 8:00 am to 5:00 pm, Monday - Friday; and on-site service assistance from 8:00 am to 5:00 pm, Monday - Friday; and Solaris maintenance releases. This service provides live telephone transfer of software fixes and 4 hour on-site response for urgent problems.

All Sun hardware has a one year warranty. During the warranty period, the monthly price for the Silver Program is 50% of the usual monthly price.

8.2.2 Sybase Standard Technical Support

Sybase Standard Technical Support includes:

- Product updates.
- Unlimited, toll-free telephone service to assist in product installation, syntax, and usage that is available from 6:00 a.m. to 5:00 p.m., Monday through Friday.

8.3 Discounts

The following generally available discounts to any buyer with like conditions were applied to the priced configurations:

- a 15% Sun support prepayment discount

The tested configuration used one LAN for connecting the driver system to the client system and the server system.

The target system is priced with one LAN between the client machine and the server. There are 3 LAN's between the client and the workstation "terminals". There were 386 workstations "terminals" on two fo the LANs and 388 on the third.. This approach was used in a prior published TPC-C version 3.0 FDR. By limiting the number of workstation "terminals" to fewer than 500 per LAN, less than 14% of the LAN bandwidth was used under full load. This approach was approved by the benchmark auditor.

7.5 WAN/LAN Bandwidth

The bandwidth of the network(s) used in the tested/priced configuration must be disclosed.

Local area networks (LANs) with a bandwidth of ten (10) megabits per second were used in the tested / priced system.

7.6 Operator Intervention

If the configuration requires operator intervention, the mechanism and the frequency of this intervention must be disclosed.

The Ultra Enterprise 150 configuration reported does not require any operator intervention to sustain the reported throughput.

8 - Clause 7 Related Items

8.1 System Pricing

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

```
# Enter the following parameters related to tpcc_master.com

set users = ( 1160 )   # Number of slaves on each machine
set rte_machines = ( toners ) # Names of rte machines
set clnt_machines = ( playwright ) # Names of client machines (same # as #rtes)
```

The code used to generate the transactions and record response times is shown in Appendix E.

7.2 Emulated Components

It must be demonstrated that the functionality and performance of the components being emulated in the Driver System are equivalent to that of the priced system. The results of the test described in Clause 6.6.3.4 must be disclosed.

In the priced configuration, workstations are connected to the clients via telnet in the same way as the emulated system. In the tested configuration, the driver system emulates the workstations by making a direct connection to the SUT for each terminal.

7.3 Configuration Diagrams

A complete funcitional diagram of both the benchmark configuration and the configuration of the proposed (target) system must be disclosed. A detailed list of all software and hardware functionality being performed on the Driver System, and its interface to the SUT must be disclosed (see Clause 6.6.3.6).

Figure 1 is a diagram of the benchmarked configuration and Figure 2 shows the configuration of the priced configuration. Section 1.4 of this Full Disclosure Report gives details on both configurations.

7.4 Network Configuration

The network configurations of both the tested services and the proposed (target) services which are being represented and a thorough explanation of exactly which parts of the proposed configuration are being replaced with the Driver System must be disclosed (see Clause 6.6.4).

6.20 Deliverys Skipped

The percentage of Delivery transactions skipped due to there being fewer than necessary orders in the New-Order table must be disclosed.

See Table 1 for results.

6.21 Checkpoints

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

One checkpoint occurred at 280 seconds after the start of ramp-up and one at 480 seconds after the start of the measurement interval. The interval between the two checkpoints was 30 minutes.

7 - Clause 6 Related Items

7.1 RTE Description

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

The RTE used was developed by Sun Microsystems Computer Company and is proprietary. It consists of a *master_rte* program which forks off the individual RTE processes and controls the run. After the run completes, a separate report generator program collects all the log files and generates the final statistics of a run.

Inputs to the RTE include the names of the RTE machines to run on, client machines to attach to, the database scale, the ramp-up, measurement and ramp-down times. The script used to set these values is shown below:

```
setenv driver   tpcc_master.com        #Name of driver script
setenv  ramp_up         1560    # ramp_up interval (secs)
setenv  stdy_state      1800    # steady-state/measurement interval (secs)
setenv  ramp_down       120     # ramp_down interval (secs)
setenv  trigger_time    2450    # Trigger time for users to login
setenv  scale           116     # # of warehouses
```

6.14 Numerical Results

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

See Table 1 for results.

6.15 New-Orders Rolled-Back

The percentage of New-Order transactions rolled back as a result of invalid item number must be disclosed.

See Table 1 for results.

6.16 Order-Line Average

The average number of order-lines entered per New-Order transaction must be disclosed.

See Table 1 for results.

6.17 Remote Order-Lines

The percentage of remote order-lines entered per New-Order transaction must be disclosed.

See Table 1 for results.

6.18 Remote Payments

The percentage of remote payment transactions must be disclosed.

See Table 1 for results.

6.19 Customer Lastname

The percentage of customer selections by customer last name in the Payment and Order-Status transactions must be disclosed.

See Table 1 for results.

6.10 Work Performed During Steady State

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

6.10.1 Checkpoint

A Sybase SQL Server 11.0 checkpoint writes all buffers in memory to disk so that data on disk matches what is in memory. Checkpoints are marked by a special record written into the logs. One checkpoint was implemented in the measurement run.

6.11 Reproducibility

A description of the method used to determine the reproducibility of the measurement results must be reported.

In a repeat run, a throughput of 1321.87 tpmC was achieved.

6.12 Measurement Period Duration

A statement of the duration of the measurement interval for the reported Maximum Qualified Throughput (tpmC) must be included.

The measurement interval was 30 minutes. There was 1 checkpoint during the measurement interval.

6.13 Transaction Mix Regulation

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

The weighted distribution algorithm as described in Clause 5.2.4.1 of the TPC-C specification was used to regulate the transaction mix. Weights for the various transactions were statically assigned.

6.8 Throughput versus Elapsed Time

A graph of throughput versus elapsed time (see Clause 6.6.5) must be reported for the New-Order transaction.

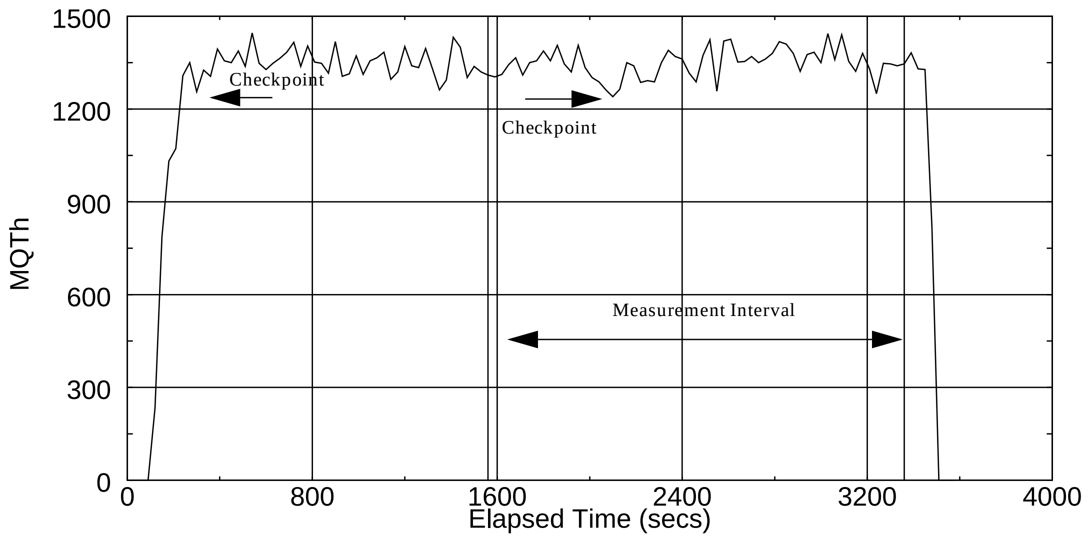


Figure 21: Throughput versus Time

6.9 Steady State Determination

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

The transaction throughput rate (tpmC) and response times were relatively constant after the initial 'ramp up' period. The throughput and response time were verified by examining the throughput (tpmC) graph reported at 30 second intervals for the duration of the benchmark. Ramp up, steady state, and ramp down are clearly discernible in the graph, Figure 21.

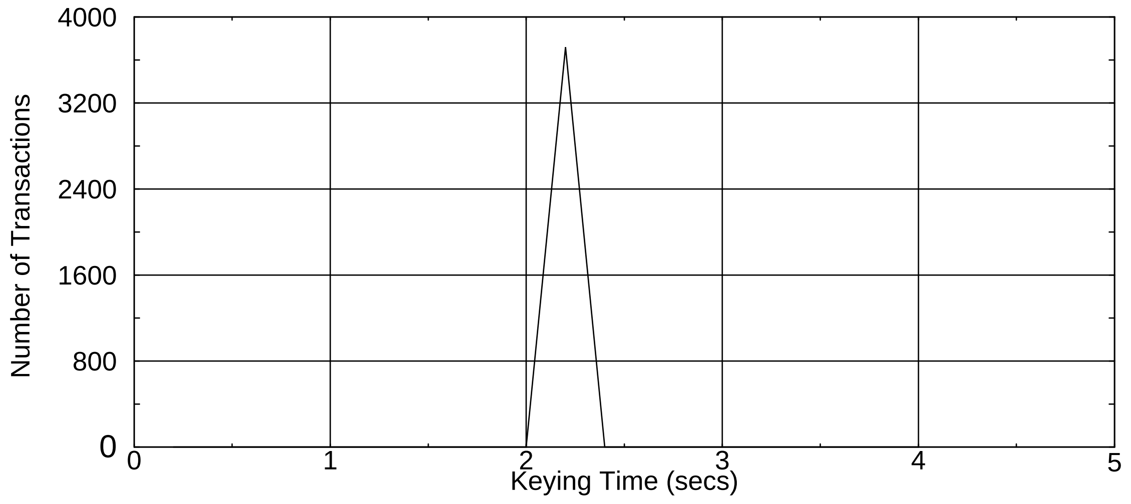


Figure 19: Delivery Keying Time Distribution

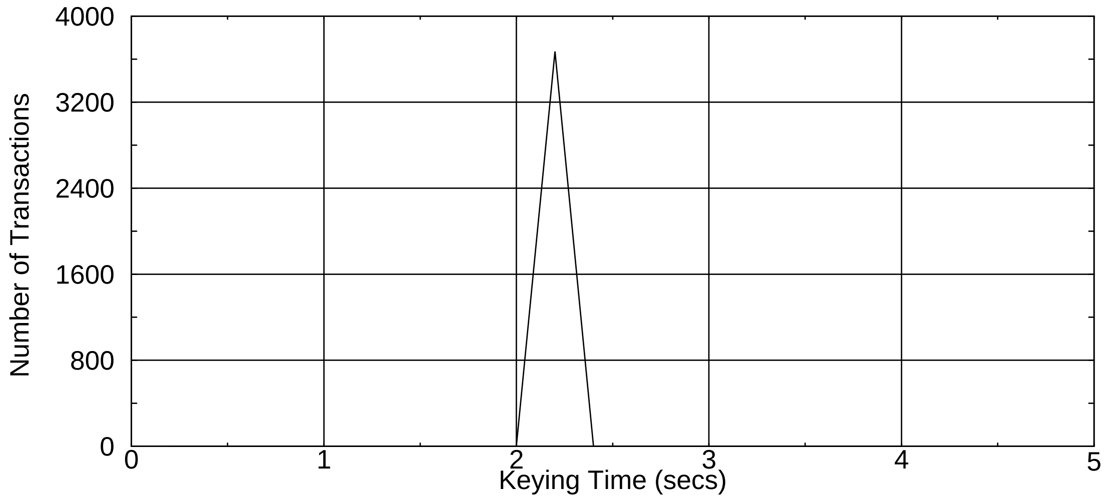


Figure 20: Stock Level Keying Time Distribution

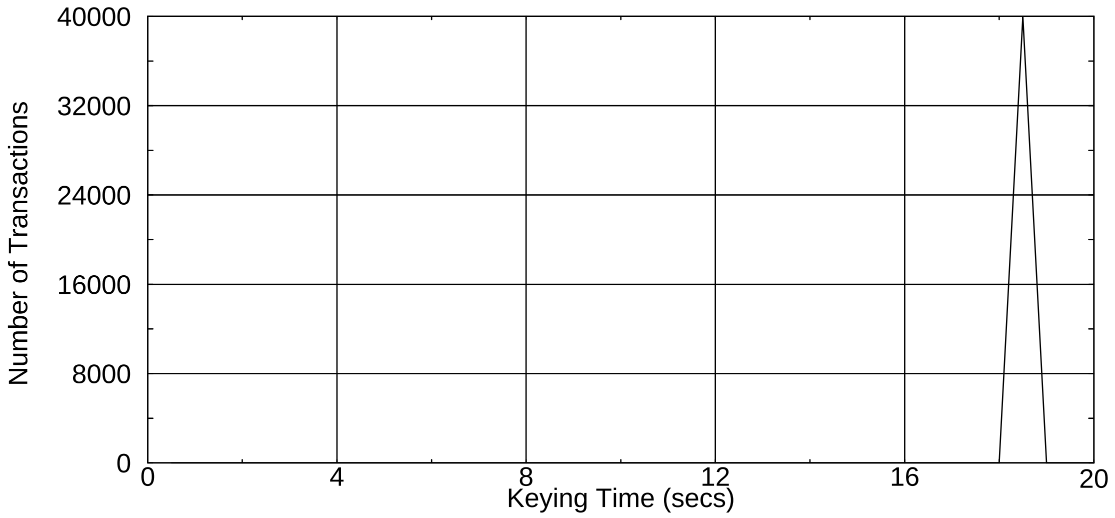


Figure 17: Payment Keying Time Distribution

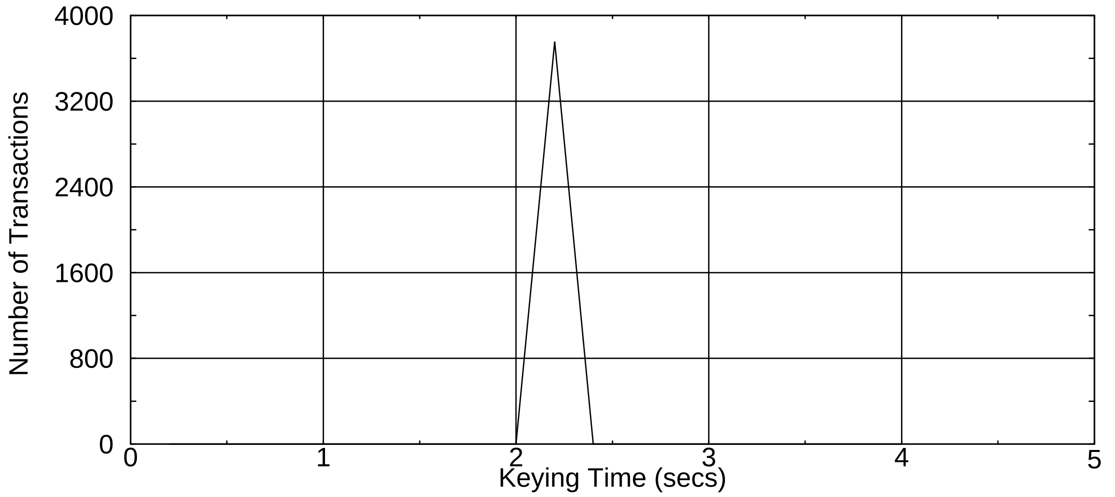


Figure 18: Order Status Keying Time Distribution

6.7 Keying Time Frequency Distribution Curves

Keying Time frequency distribution curves (see Clause 5.6.4) must be reported for each transaction type.

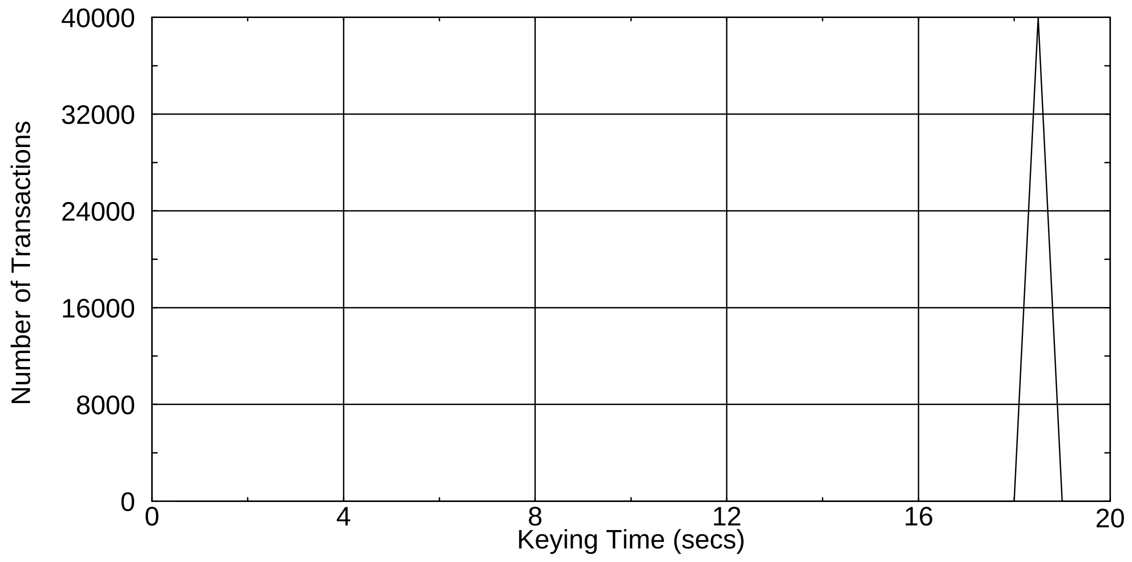


Figure 16: New Order Keying Time Distribution

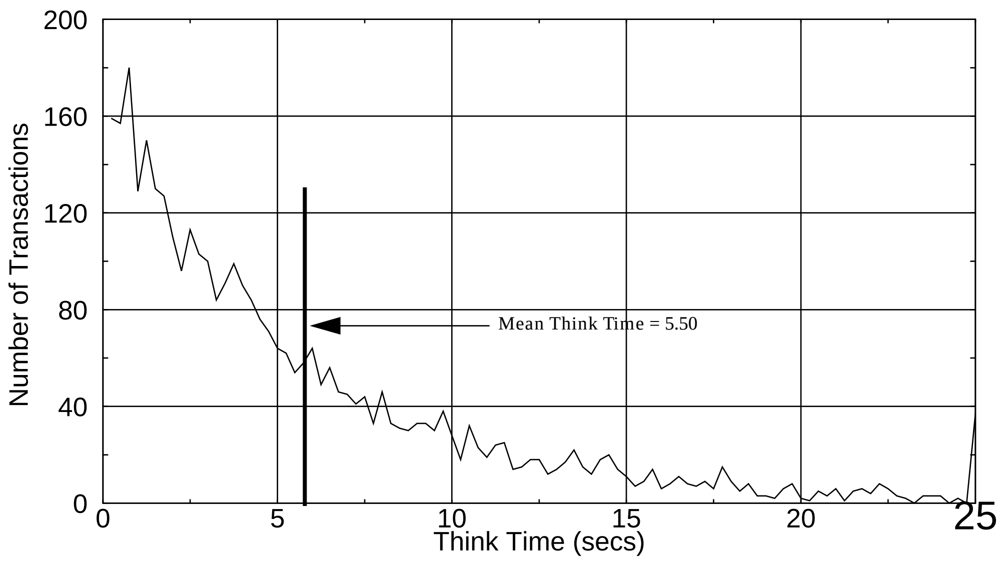


Figure 15: Stock Level Think Time Distribution

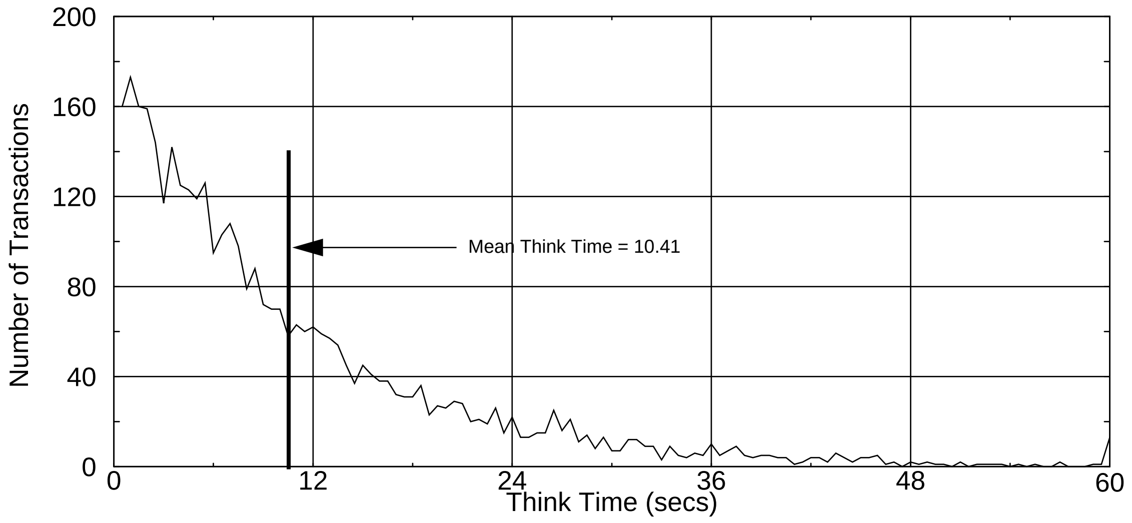


Figure 13: Order Status Think Time Distribution

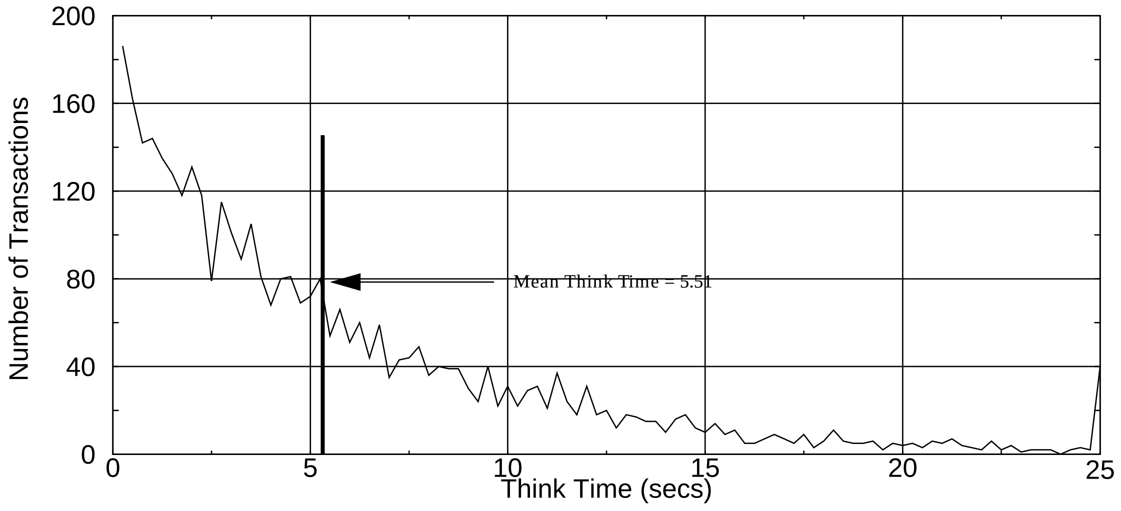


Figure 14: Delivery Think Time Distribution

6.6 Think Time distribution curves

Think Time frequency distribution curves (see Clause 5.6.3) must be reported for each transaction type.

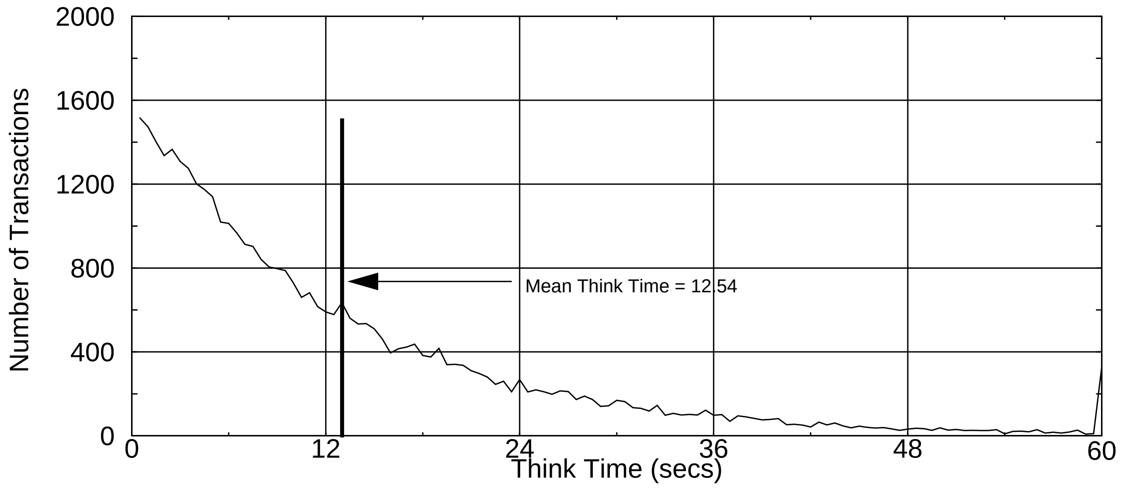


Figure 11: New Order Think Time Distribution

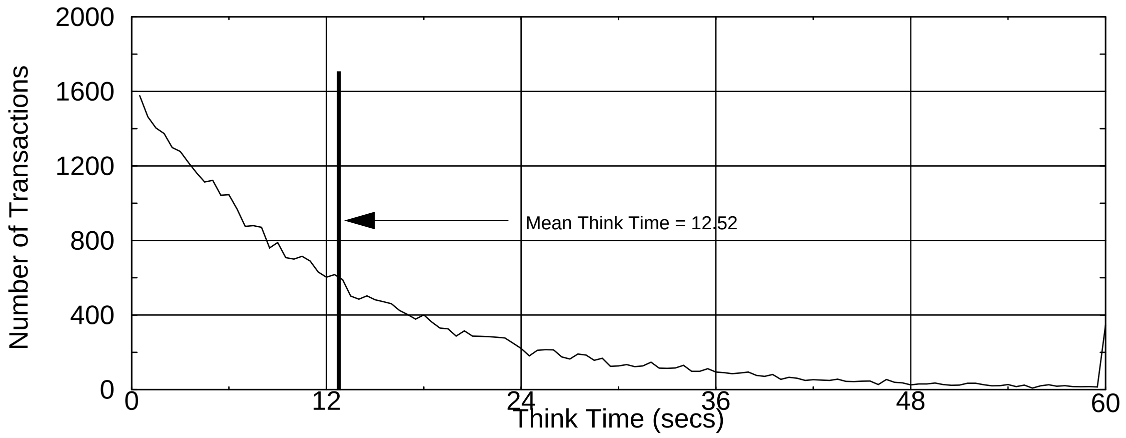


Figure 12: Payment Think Time Distribution

6.5 Response time versus throughput

The performance curve for response times versus throughput (see Clause 5.6.2) must be reported for the New Order transaction.

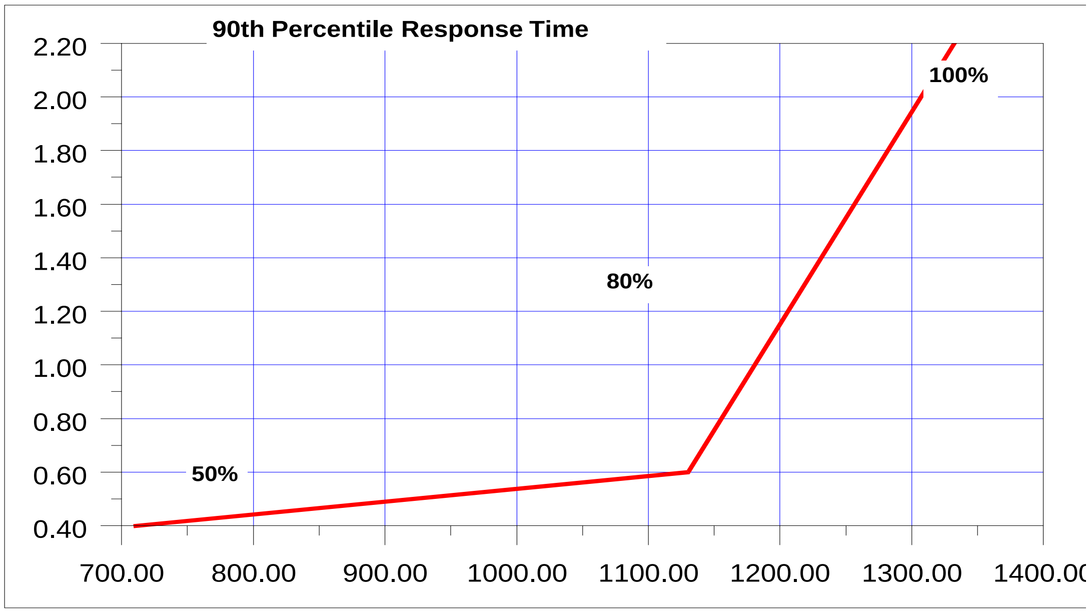


Figure 10: Response Time versus Throughput

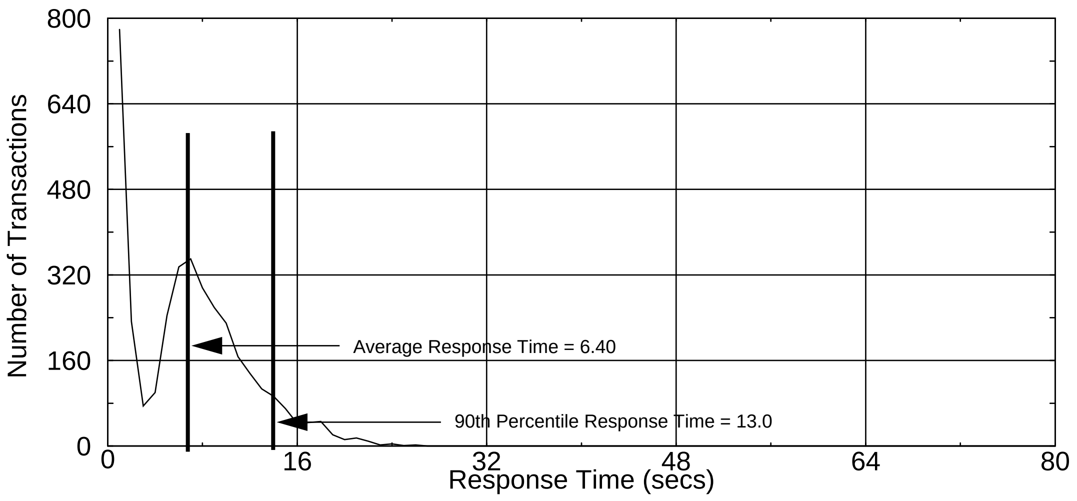


Figure 9: Stock Level Response Time Distribution

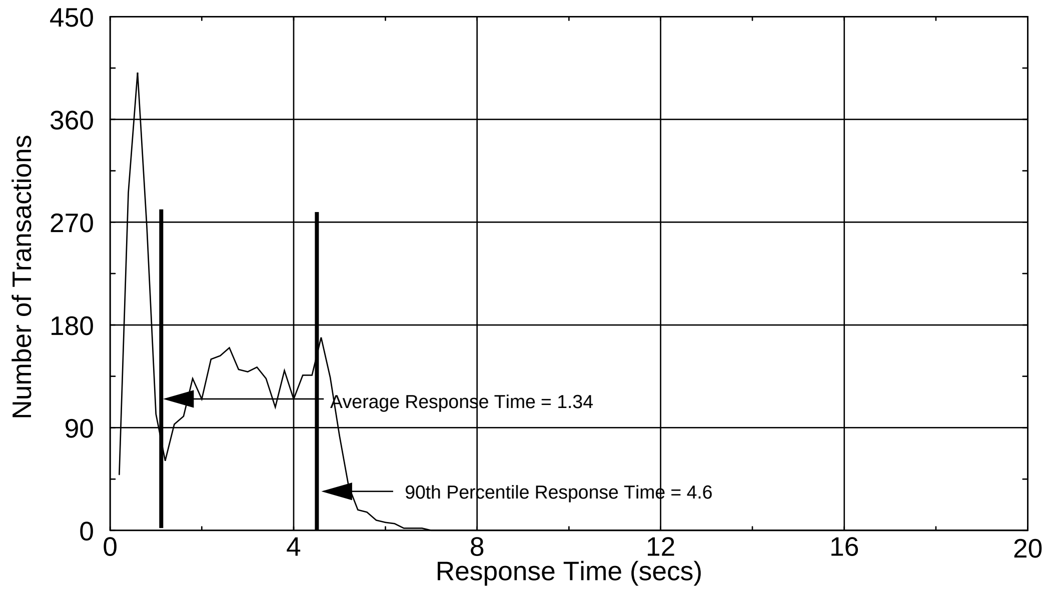


Figure 7: Order Status Response Time Distribution

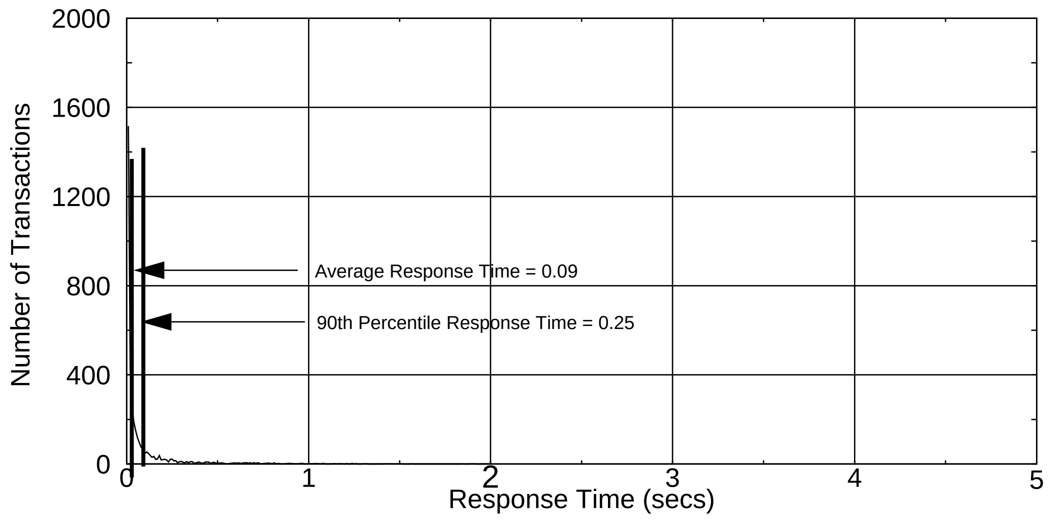


Figure 8: Delivery Response Time Distribution

6.4 Response Time Frequency Distribution Curves

Response Time frequency distribution curves (see Clause 5.6.1) must be reported for each transaction type.

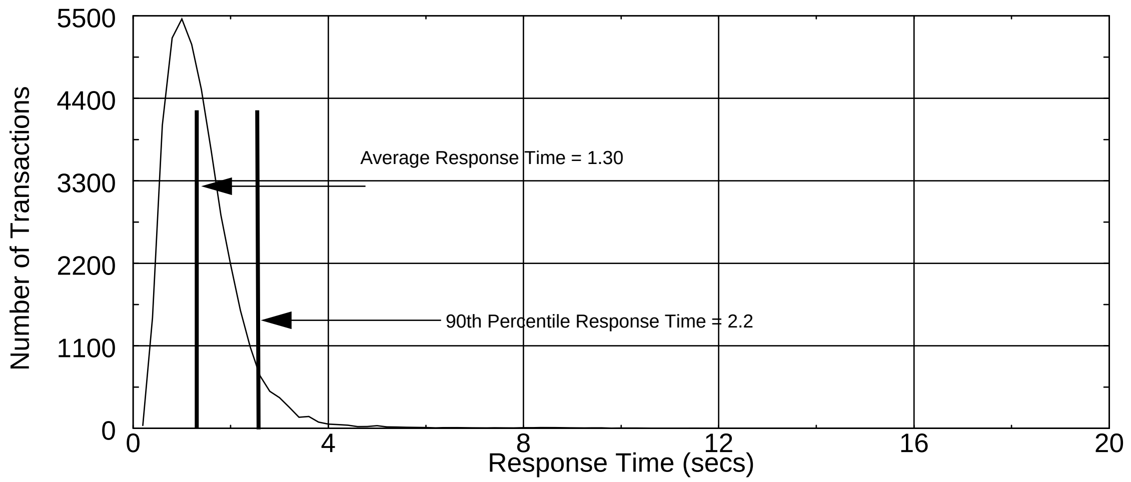


Figure 5: New Order Response Time Distribution

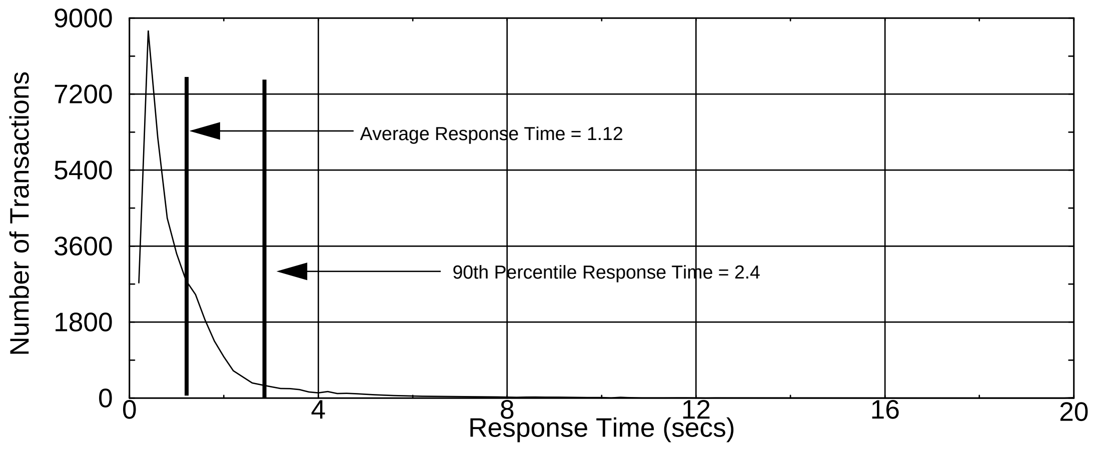


Figure 6: Payment Response Time Distribution

6.3 Keying and Think Times

The minimum, the average, and the maximum keying and think times must be reported for all transaction types.

Table 4: Keying Times

Type	Average	Maximum	Minimum
New-Order	18.01	18.08	18.0
Payment	3.01	3.04	3.00
Order-Status	2.01	2.02	2.00
Interactive Delivery	2.01	2.03	2.00
Stock-Level	2.01	2.09	2.00

Table 5: Think Times

Type	Average	Maximum	Minimum
New-Order	12.54	125.00	0.00
Payment	12.52	125.00	0.00
Order-Status	10.41	85.56	0.07
Interactive Delivery	5.51	43.02	0.00
Stock-Level	5.50	48.39	0.00

6 - Clause 5 Related Items

6.1 Measured tpmC

Measured tpmC must be reported.

The measured tpmC was *1332.50*

6.2 Response Times

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

Table 3: Response Times

Type	Average	Maximum	90% percentile
New-Order	1.30	14.62	2.20
Payment	1.12	27.32	2.40
Order-Status	2.38	6.64	4.60
Interactive Delivery	0.09	2.34	0.26
Deferred Delivery	5.14	24.00	11.00
Stock-Level	6.40	25.28	13.00
Menu	0.09	1.65	0.50

5.3 Type of Database

A statement must be provided that describes:

1. The data model implemented by the DBMS used (e.g., relational, network hierarchical).

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

Sybase SQL Server 11.0 is a relational database management system.The interface we used was Sybase DB-Library and stored procedures.

5.4 Mapping of Database

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

No table partitioning or replication was done.

5.5 180 Day Space Computation

Details of the 180 day space computations along with proof that the database is configured to sustain 8 hours of growth for the dynamic tables (Order, Order-Line, and History) must be disclosed (see Clause 4.2.3).

The 180 day space computation is shown in Appendix D.

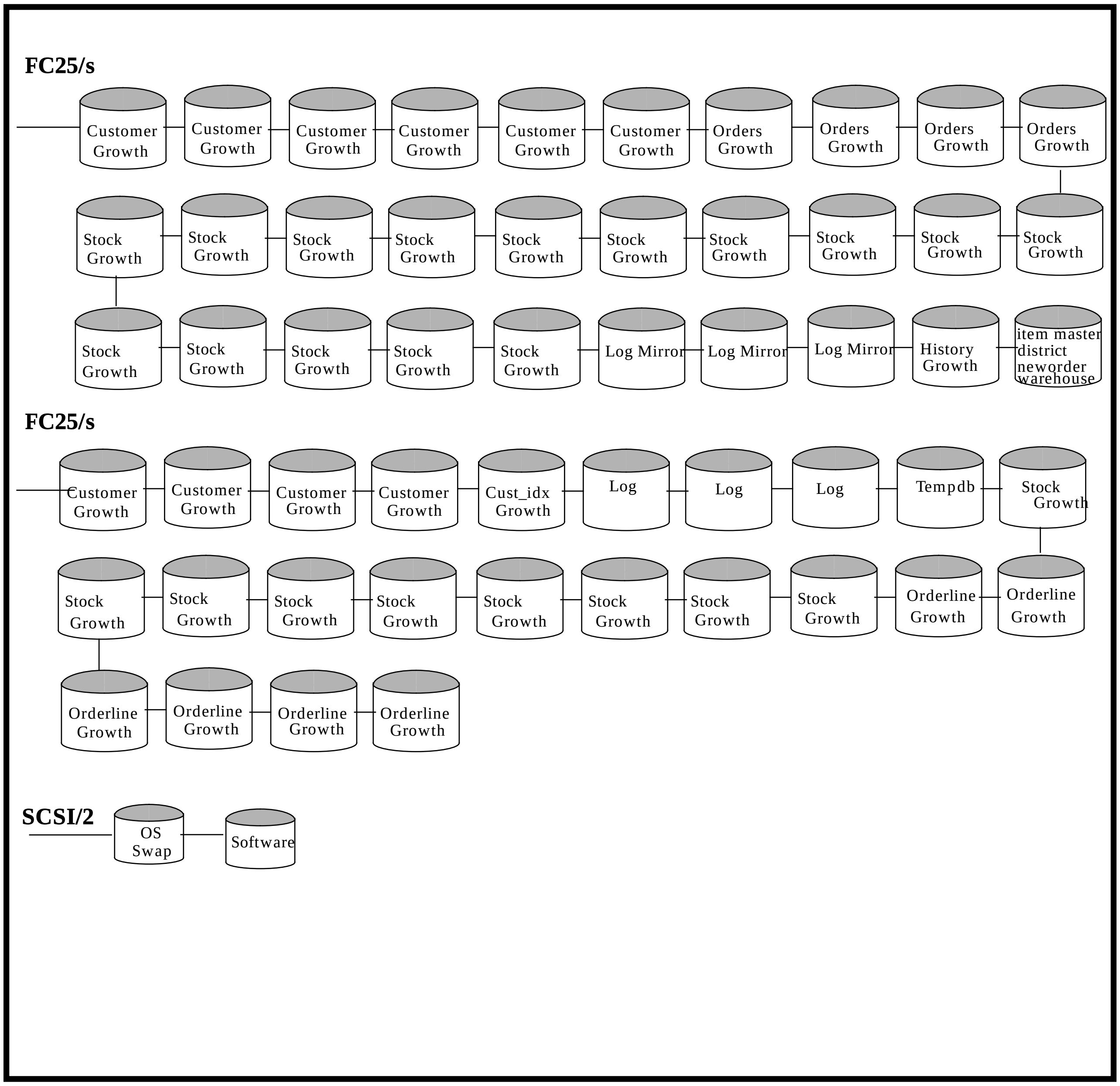


Figure 4: Database Layout of the Priced System

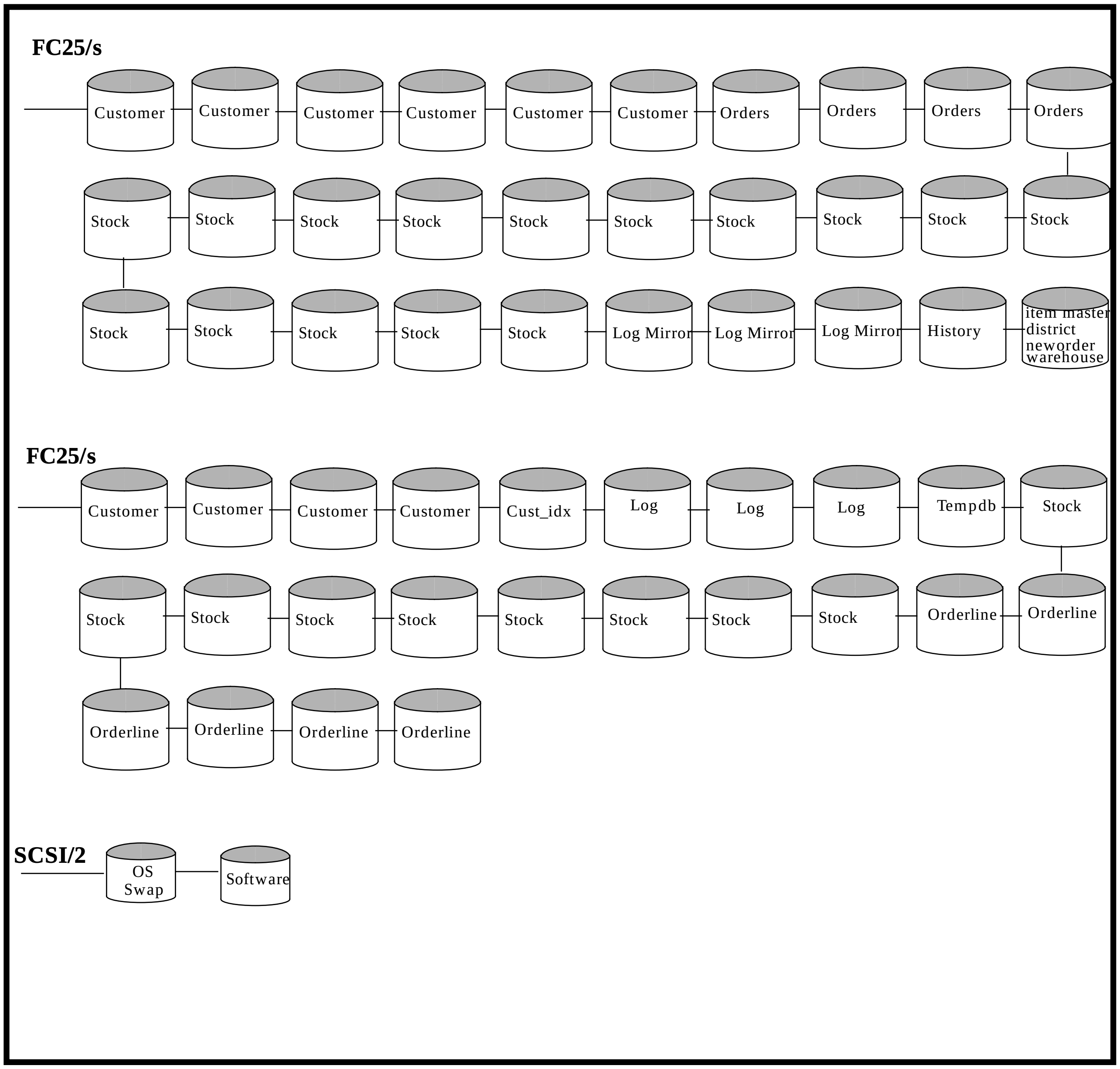


Figure 3: Database Layout of the Tested System

Table 2: Cardinality of Tables

Table	Occurrences
Warehouse	116
District	1160
Customer	3480000
History	3480000
Orders	3480000
New order	1044000
Order line	34800000
Stock	11600000
Item	100000

5.2 Database Layout

The distribution of tables and logs across all media must be explicitly depicted for the tested and priced systems.

Figure 3 depicts the distribution of the database for the tested system. Figure 4 depicts the distribution of the database for the priced system.

2. A fully-scaled test was executed. On the driver system, the committed and rolled back New-Order transactions were recorded in a “success” file.

3. After 5 minutes into the measurement period, the SUT’s primary power was removed.

4. The test was aborted on the driver.

5. Power was restored to the SUT and a normal system recovery was done. A recovery was automatically performed by Sybase SQL Server 11.0 when the database was restarted and brought on-line. The recovery restored the database to the consistent point just after the last committed transaction had occurred before the induced failure.

6. The contents of the “success” file on the driver and the ORDERS table were compared to verify that records in the “success” file for committed New-Order transactions had corresponding records in the ORDERS table. The number of transactions missed “in flight” were less than the number of users.

7. Step 1 was repeated to determine the total number of orders (count2). Count2-count1 was compared with the number of committed records in the “success” file.

5 - Clause 4 Related Items

5.1 Initial Cardinality of Tables

The Cardinality (e.g. number of rows) of each table, as it existed at the start of the benchmark run (see Clause 4.2), must be disclosed. If the database was over-scaled and inactive rows of the WAREHOUSE table were deleted (see Clause 4.2.2) the cardinality of the WAREHOUSE table as initially configured and the number of rows deleted must be disclosed.

The TPC-C database for this test was configured with 116 warehouses.

the TPC-C specifications defines the isolation requirements which must be met by the TPC-C transactions. Sufficient conditions must be enabled at either the system or application level to ensure the required isolation is maintained.

Tests waived by auditor as previously performed.

4.5 Durability

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

List of single failures:

> *Permanent irrecoverable failure of any single durable medium containing TPC-C database tables or recovery log data.*
>
> *Instantaneous interruption (system crash/system hang) in processing which requires system reboot to recover.*
>
> *Failure of all or part of memory (loss of contents).*

Sun Microsystems executed one durability test to satisfy the durability requirements for this implementation of TPC Benchmark C. The combined test for loss of memory and instantaneous interruption was performed with a fully scaled database under the full load of terminals. The other tests were waived by auditor as previously performed

4.5.1 Permanent Irrecoverable Failure

The loss of data disk and loss of log disk tests were waived by auditor as previously performed.

4.5.2 Instantaneous Interruption and Loss of Memory

Instantaneous interruption and loss of memory tests were combined because the loss of power erases the contents of memory. This failure was induced by removing the SUT's primary power while the benchmark was running.

1. The D_NEXT_O_ID fields for all rows in district table were summed up to determine the initial count of the total number of orders (count1).

4 - Clause 3 Related Items

4.1 Transaction System Properties (ACID)

The results of the ACID tests must be disclosed along with a description of how the ACID requirements were met. This includes disclosing which case was followed for the execution of Isolation Test 7.

The TPC Benchmark C Standard Specification defines a set of transaction processing system properties that a system under test (SUT) must support during the execution of the benchmark. Those properties are Atomicity, Consistency, Isolation, and Durability (ACID). This section defines each of these properties, describes the steps taken to ensure that they were present during the test and describes a series of tests done to demonstrate compliance with the standard.

4.2 Atomicity

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

Tests waived by auditor as previously performed.

4.3 Consistency

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

Tests waived by auditor as previously performed.

4.4 Isolation

Isolation can be defined in terms of phenomena that can occur during the execution of concurrent transactions. These phenomena are P0 (“Dirty Write”), P1 (“Dirty Read”), P2 (“Non-repeatable Read”) and P3 (“Phantom”). The table in Clause 3.4.1 of

3.5 Transaction Statistics

Table 1 lists the numerical quantities that Clauses 8.1.3.5 to 8.1.3.11 requires.

Table 1: Transaction Statistics

Transaction Type	Statistics	Percentage
New Order	Home warehouse	99.01
	Remote warehouse	0.99
	Rolled back transactions	1.06
	Average items per order	9.98
Payment	Home warehouse	84.03
	Remote warehouse	15.07
	Non-primary key access	59.65
Order Status	Non-primary key access	59.27
Delivery	Skipped transactions	0.00
Transaction Mix	New order	44.21
	Payment	43.46
	Order status	4.16
	Delivery	4.11
	Stock level	4.06

3.6 Queueing Mechanism

The queueing mechanism used to defer the execution of the Delivery transaction must be disclosed.

Delivery transactions were submitted to servers using the same Tuxedo call mechanism that other transactions used. The only difference was that the call was asynchronous - i.e., control returned to the client process immediately and the deferred delivery completed asynchronously.

3 - Clause 2 Related Items

3.1 Random Number Generation

The method of verification for the random number generation must be described.

The Random Number Generator used was the one that appeared in the article titled “Random Number Generators: Good Ones Are Hard To Find” in the communications of the ACM - October 1988, Volume 31, Number 10. The properties of this random number generator are well-known and are documented in the article as producing a uniformly distributed pseudo-random sequence. To generate a random number, the driver programs first use a seed based on the host address, current time and the process-id of the respective session. This guarantees that each emulated user on all the RTE machines is mathematically independent of others.

3.2 Input/Output Screen Layouts

The actual layout of the terminal input/output screens must be disclosed.

The screen layouts are shown in Appendix F.

3.3 Terminal Feature Verification

The method used to verify that the emulated terminals provide all the features described in Clause 2.2.2.4 must be explained.

The terminal features were verified by manually exercising each specification on a representative Ultraserver 1 running Solaris 2.5.

3.4 Presentation Manager or Intelligent Terminal

Any usage of presentation managers or intelligent terminals must be explained.

The TPC-C forms module was implemented using the standard System V curses libraries.

2.5 Table Replication

Replication of tables, if used, must be disclosed (see Clause 1.4.6).

No tables were replicated in this implementation.

2.6 Table Attributes

Additional and/or duplicated attributes in any table must be disclosed along with a statement on the impact on performance (see Clause 1.4.7).

No additional or duplicate attributes were added to any of the tables.

2 - Clause 1 Related Items

2.1 Table Definitions

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

Appendix B describes the programs that define, create, and populate a Sybase SQL Server 11.0 database for TPC-C testing.

2.2 Physical Organization of Database

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

Space was allocated to Sybase SQL Server 11.0 on the server according to the data in section 5.2. The size of the database devices on each disk drive was calculated to provide even distribution of load across the disk drives.

2.3 Insert and Delete Operations

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

There were no restrictions on insert and delete operations to any tables beyond the limits defined in Clause 1.4.11..

2.4 Partitioning

While there are a few restrictions placed upon horizontal or vertical partitioning of tables and rows in the TPC-C benchmark (see Clause 1.6), any such partitioning must be disclosed.

Partitioning was not used in this implementation.

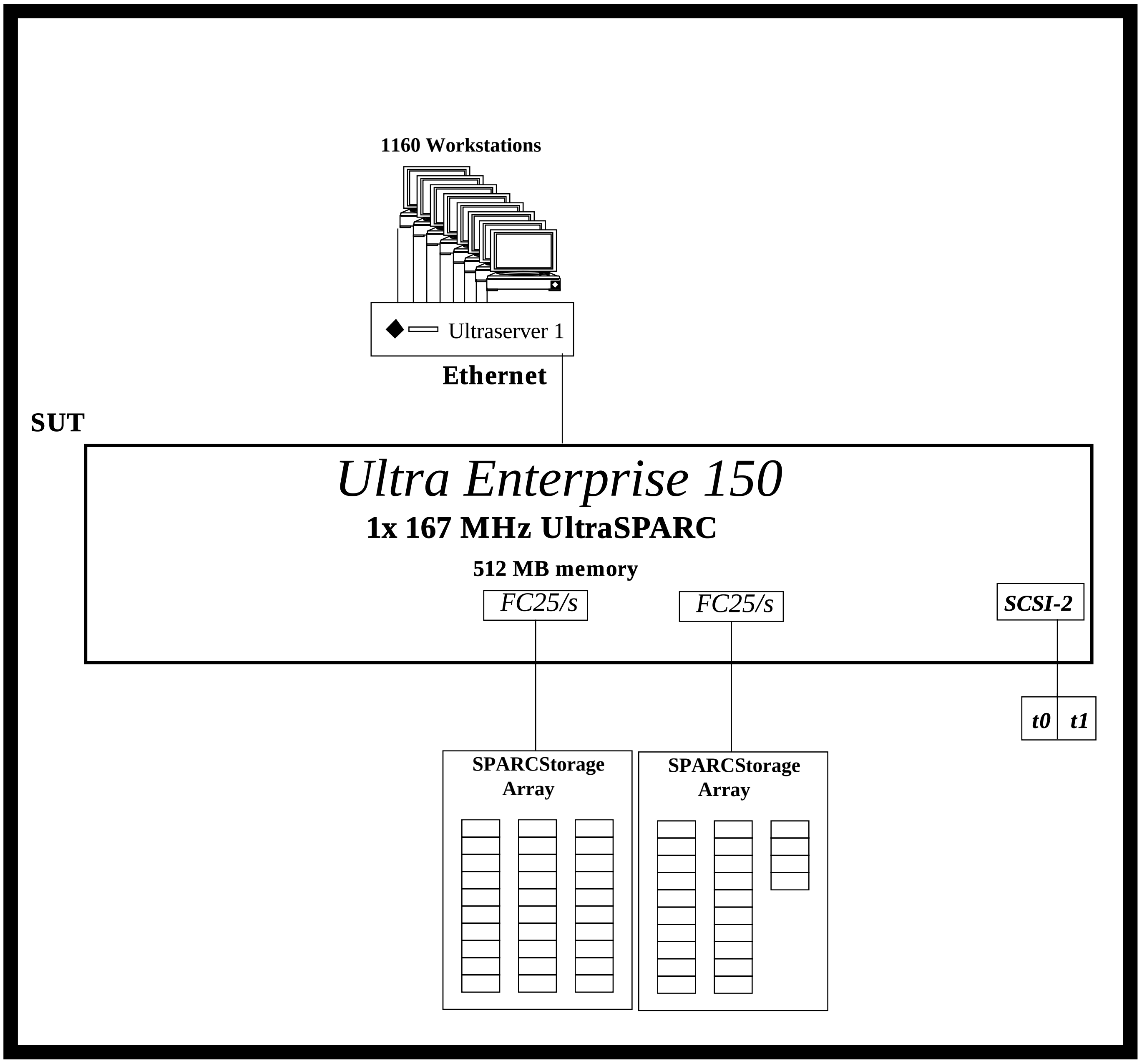


Figure 2: The Sun Ultra Enterprise 150 Priced Configuration

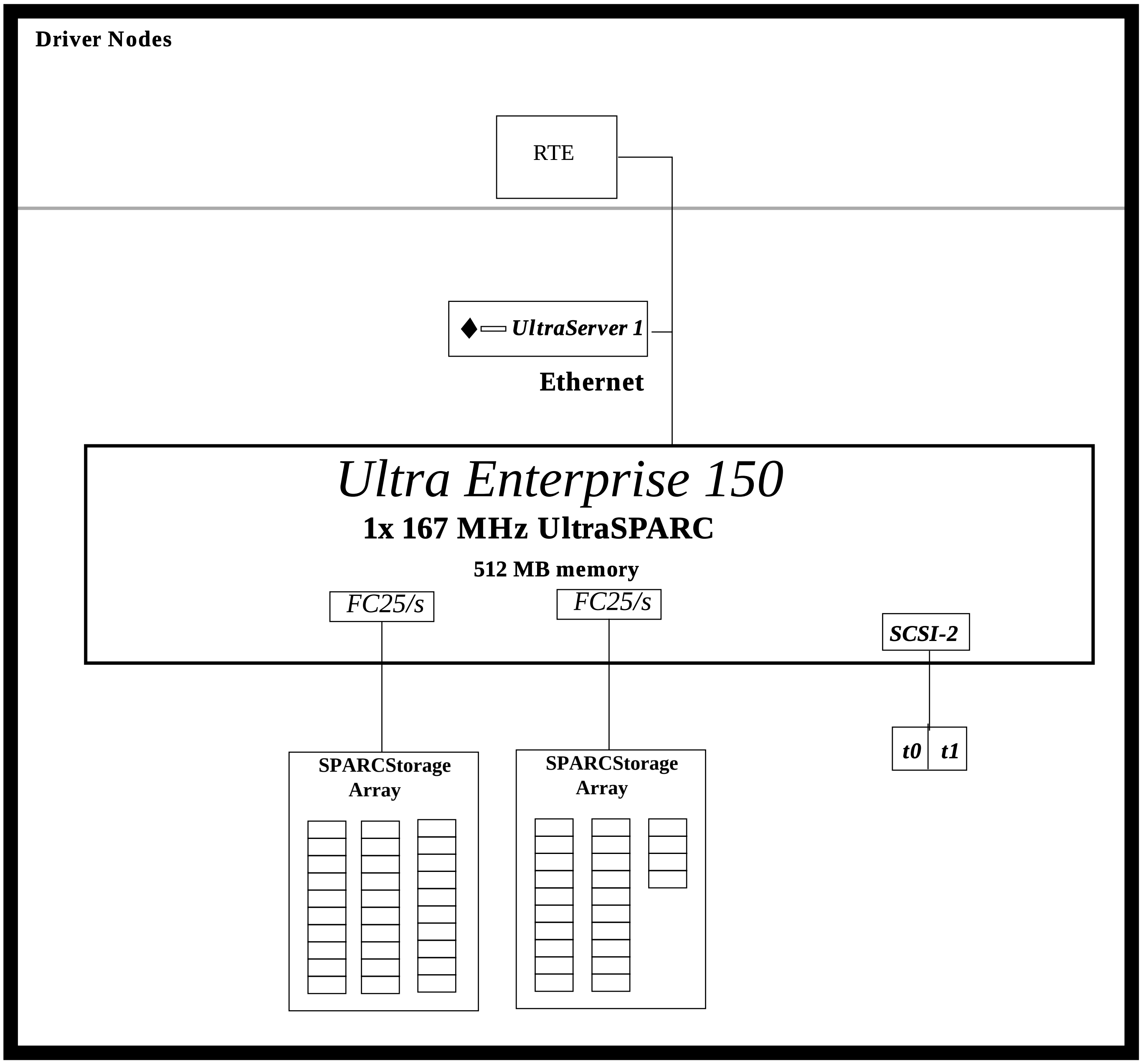


Figure 1: The Sun Ultra Enterprise 150 Benchmark Configuration

The benchmark configuration used a Remote Terminal Emulator (RTE) to emulate TPC-C user sessions. The driver systems were directly connected through ethernet to the Ultraserver 1 Model 140 which emulated the database client sessions.

1.4 Configuration Diagrams

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

Figure 1 is a diagram of the benchmark configuration. Figure 2 is a configuration diagram of the priced system.

Configuration Items for the Ultra Enterprise 150

For the priced configuration, the server machine was a Ultra Enterprise 150 which consisted of the following:

- One UltraSPARC 167 MHz Processor.
- 512 MB of main memory.
- One Internal SCSI-2 controller
- Two Internal 2.1 GB SCSI disks.
- One SPARCstorage Array Model 112 (30 x 2.1GB SCSI disks).
- One SPARCstorage Array Model 112 (24 x 2.1GB SCSI disks).
- Two Fiber Channel host adaptors.
- CD-ROM
- 5 GB Backup Tape Device.

For the benchmark configuration, we used the same configuration as above except we used two SPARCstorage Array Model 102.

For both the priced and benchmark configurations, the client machine was an UltraServer 1 Model 140 contained:

- One UltraSPARC 143 MHz Processor.
- 512 MB of Main Memory.
- One Internal SCSI-2 controller.
- One Internal 2.1 GB SCSI disk.
- CD-ROM.

1- General Items

1.1 Application Code and Definition Statements

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

Appendix A contains the application source code that implements the transactions and forms modules.

1.2 Sponsor

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

This benchmark test was sponsored by Sun Microsystems Computer Company and Sybase Inc.

1.3 Parameter Settings

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

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

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

Appendix C contains all the required parameter settings.

TPC Benchmark™C Full Disclosure

Introduction

The *TPC Benchmark ™C Standard Specification* requires test sponsors to publish, and make available to the public, a full disclosure report for the results to be considered compliant with the Standard. The required contents of the full disclosure report are specified in Clause 8.

This report is intended to satisfy the Standard's requirement for full disclosure. It documents the compliance of the benchmark tests reported in the *TPC Benchmark ™C* results for the Sun Microsystems Ultra Enterprise 150 running the Sybase SQL Server 11.0 RDBMS.

In the *Standard Specification*, the main headings in Clause 8 are keyed to the other clauses. The headings in this report use the same sequence, so that they correspond to the titles or subjects referred to in Clause 8.

Each section in this report begins with the text of the corresponding item from Clause 8 of the *Standard Specification*, printed in italic type. The plain type text that follows explains how the tests comply with the TPC C™ Benchmark requirement. In sections where Clause 8 requires extensive listings, the section refers to the appropriate appendix at the end of this report.

Additional Copies

To request additional copies of this report, write to the following address:

TPC Benchmark Reports
Sun Microsystems. Inc.
2550 Garcia Ave.
Mountain View, CA, 94043

Attn.: Rod Tansimore
Mailstop: UPAL1-324

Or, copies may be ordered from the administrator of the TPC:

Shanley P.R.
777 N First Street, Suite 600
San Jose, CA 95112-6311
(408) 295-8894
FAX (408) 295-2613

Preface

This report documents the compliance of the Sun Microsystems TPC Benchmark ™C testing on the Ultra Enterprise 150 running Sybase SQL Server 11.0 RDBMS with the *TPC Benchmark* ^{TM}*C Standard Revision 3.0*.

These tests were run using the Sybase SQL Server 11.0 RDBMS running with Solaris 2.5 and iTi Tuxedo 4.2.1 on the Ultra Enterprise 150.

Document Structure

The *TPC Benchmark ™C Full Disclosure Report* is organized as follows:

- The main body of the document lists each item in Clause 8 of the TPC Benchmark ™C Standard and explains how each specification is satisfied.
- Appendix A contains the application source code that implements the transactions and forms modules.
- Appendix B contains the code used to create and load the database.
- Appendix C contains the configuration information for the operating system, the RDBMS and Tuxedo.
- Appendix D contains the 180-day space calculations.
- Appendix E contains the code used to generate transactions and measure response times.
- Appendix F contains the screen layouts of all the forms.
- Appendix G contains all the price quotes.

Tables

Table 1: Transaction Statistics 10

Table 2: Cardinality of Tables 14

Table 3: Response Times. 18

Table 4: Keying Times. 19

Table 5: Think Times. 19

Figure 17: Payment Keying Time Distribution 28
Figure 18: Order Status Keying Time Distribution 28
Figure 19: Delivery Keying Time Distribution 29
Figure 20: Stock Level Keying Time Distribution 29
Figure 21: Throughput versus Time 30

Figures

Figure 1: The Sun Ultra Enterprise 150 Benchmark Configuration 5

Figure 2: The Sun Ultra Enterprise 150 Priced Configuration 6

Figure 3: Database Layout of the Tested System 15

Figure 4: Database Layout of the Priced System 16

Figure 5: New Order Response Time Distribution 20

Figure 6: Payment Response Time Distribution 20

Figure 7: Order Status Response Time Distribution 21

Figure 8: Delivery Response Time Distribution 21

Figure 9: Stock Level Response Time Distribution 22

Figure 10: Response Time versus Throughput 23

Figure 11: New Order Think Time Distribution 24

Figure 12: Payment Think Time Distribution 24

Figure 13: Order Status Think Time Distribution 25

Figure 14: Delivery Think Time Distribution 25

Figure 15: Stock Level Think Time Distribution 26

Figure 16: New Order Keying Time Distribution 27

7.3 Configuration Diagrams 34

7.4 Network Configuration 34

7.5 WAN/LAN Bandwidth 35

7.6 Operator Intervention 35

8 - Clause 7 Related Items 35

8.1 System Pricing. 35

8.2 Support Pricing 36

8.2.1 Sun Hardware and Software Support . . 36

8.2.2 Sybase Standard Technical Support 36

8.3 Discounts 36

8.4 Availability. 37

8.5 TpmC, Price/TpmC 37

9 - Clause 8 Related Items 37

9.1 Auditor’s Report. 37

A. Appendix A: Application Code 41

B. Appendix B: Database Design 79

C. Appendix C: Tunable Parameters. 97

D. Appendix D: Disk Storage. 103

E. Appendix E: Driver Scripts 107

F. Appendix F: Screen Layouts 111

G. Appendix G: Price Quotes. 115

6.2 Response Times 18
6.3 Keying and Think Times 19
6.4 Response Time Frequency Distribution Curves 20
6.5 Response time versus throughput 23
6.6 Think Time distribution curves 24
6.7 Keying Time Frequency Distribution Curves 27
6.8 Throughput versus Elapsed Time 30
6.9 Steady State Determination 30
6.10 Work Performed During Steady State 31
6.10.1 Checkpoint 31
6.11 Reproducibility 31
6.12 Measurement Period Duration 31
6.13 Transaction Mix Regulation 31
6.14 Numerical Results 32
6.15 New-Orders Rolled-Back 32
6.16 Order-Line Average 32
6.17 Remote Order-Lines 32
6.18 Remote Payments 32
6.19 Customer Lastname 32
6.20 Deliverys Skipped 33
6.21 Checkpoints 33
7 - Clause 6 Related Items 33
7.1 RTE Description 33
7.2 Emulated Components 34

3 - Clause 2 Related Items 9

3.1 Random Number Generation 9

3.2 Input/Output Screen Layouts. 9

3.3 Terminal Feature Verification 9

3.4 Presentation Manager or Intelligent Terminal 9

3.5 Transaction Statistics 10

3.6 Queueing Mechanism. 10

4 - Clause 3 Related Items 11

4.1 Transaction System Properties (ACID). 11

4.2 Atomicity 11

4.3 Consistency 11

4.4 Isolation 11

4.5 Durability. 12

4.5.1 Permanent Irrecoverable Failure. 12

4.5.2 Instantaneous Interruption and Loss of Memory. 12

5 - Clause 4 Related Items 13

5.1 Initial Cardinality of Tables 13

5.2 Database Layout 14

5.3 Type of Database. 17

5.4 Mapping of Database 17

5.5 180 Day Space Computation 17

6 - Clause 5 Related Items 18

6.1 Measured tpmC. 18

Contents

Abstract iii

1. TPC Benchmark ™C Full Disclosure. 1

Introduction 1

1- General Items 2

1.1 Application Code and Definition Statements 2

1.2 Sponsor. 2

1.3 Parameter Settings 2

1.4 Configuration Diagrams 3

2 - Clause 1 Related Items 7

2.1 Table Definitions 7

2.2 Physical Organization of Database 7

2.3 Insert and Delete Operations. 7

2.4 Partitioning 7

2.5 Table Replication 8

2.6 Table Attributes. 8

Numerical Quantities Summary for Ultra Enterprise 150

MQTH, Computed Maximum Qualified Throughput	**1332.50 tpmC**
% throughput difference, reported & reproducibility runs	< 1%

Response Times (in secs)	**90th Percentile**	**Average**	**Maximum**
Menu	0.50	0.10	1.65
New-Order	2.20	1.30	14.62
Payment	2.40	1.12	27.33
Order-Status	4.60	2.37	6.64
Delivery(interactive)	0.26	0.09	2.34
Delivery(deferred)	11.00	5.14	24.00
Stock-level	13.00	6.40	25.28

Transaction Mix, in percent of total transactions

New-Order	44.21%
Payment	43.46%
Order-Status	4.16%
Delivery	4.11%
Stock-level	4.06%

Keying/Think Times (in secs)	**Average.**	**Min.**	**Maximum**
New-Order	18.01/12.54	18.0/0	18.08/125.0
Payment	3.01/12.52	3.0/0	3.04/125.0
Order-Status	2.01/10.41	2.0/0	2.02/85.56
Delivery	2.01/5.51	2.0/0	2.03/43.02
Stock-level	2.01/5.50	2.0/0	2.09/48.39

Test Duration

Ramp-up time	26 minutes
Measurement Interval	30 minutes
Number of checkpoints	1
Checkpoint Interval	30 minutes
Number of transactions (all types) completed in measurement interval	90419

Sun Microsystems Sybase, Inc.	Ultra Enterprise 150 C/S w/1 Front-End	TPC-C Report Date: 11/18/1996

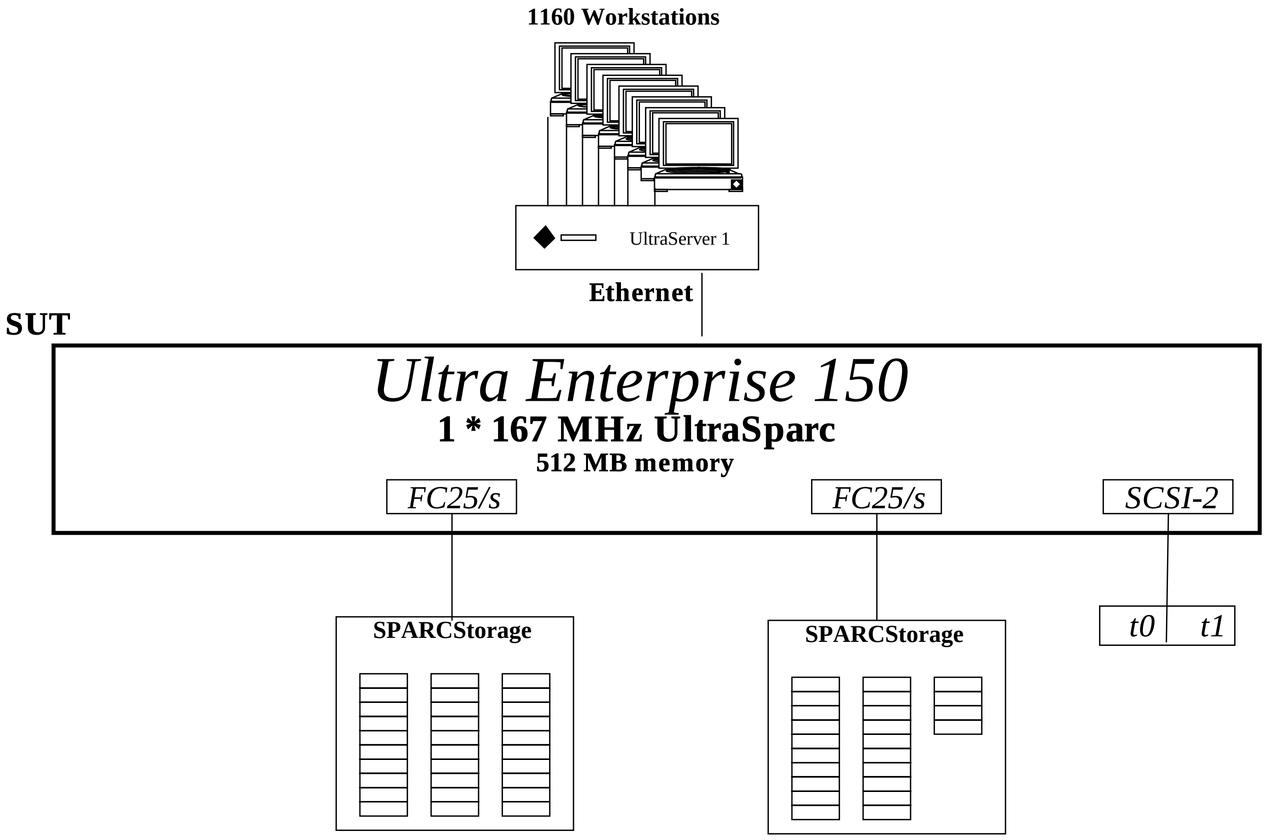
Part Number	Description	Qty	Unit Price	Extended Price	Maintenance per Month	5 Yr. Total Maintainance
Server Hardware						
Pricing from CAT Technology, Inc.						
E150-UBA1-CB	Enterprise 150 w/167 MHz UltraSPAR(	1	10496.25	10496.25	130.00	7020.00
X7003A	2*64 Mb Simm for UltraServer	4	1395.00	5580.00		0.00
X791A	SSA Model 112 w/ 18 2.1 GB Disks	1	25725.00	25725.00	120.00	6480.00
X766A	6 * 2.1 Disks for SSA	1	4800.00	4800.00		0.00
X792A	SSA Model 112 w/ 30 2.1 GB Disks	1	35100.00	35100.00	120.00	6480.00
X1057A	FibreChannel/SBus Host Adapter	2	1200.00	2400.00		0.00
X822A	4mm Tape Backup	1	1294.00	1294.00		0.00
	Ultra Enterprise 150 Subtotal:			**85395.25**		**19980.00**
Client Hardwaer						
Pricing from CAT Technology, Inc.						
A11-UAA1-9S-064CB	UltraServer 1 Model 140	1	5621.25	5621.25	76.00	4104.00
X1058A	Quad Ethernet Controler	1	746.00	746.00		0.00
X7003A	2*64 Mb Simm for UltraServer	4	1395.00	5580.00		0.00
	UltraServer 1 Subtotal:			**11947.25**		4104.00
User Connectivity						
Terminal Pricing from SunExpress, Hubs from DataComm, Inc.						
SX-WY35-G-G35	Wyse35 Terminal	2	295.00	590.00		
DEH1286	Bocca 24 Port Hubs with 10% spares	54	359.00	19386.00		
	Terminals & Connectivity Subtotal:			**19976.00**		
	ALL HARDWARE SUBTOTALS:			**117318.50**		**24084.00**
Server and Client Software						
Sun Softwrore Pricing from Sun						
SOLS-C	Solaris Software on CD-ROM	1	100.00	100.00		
CC-P	SPARC Compiler Single Usr. License	1	695.00	695.00	20.00	1140.00
	Sun Software Subtotal:	¤		**795.00**		**1140.00**
Sybase Pricing from Sybase Inc.						
SQL System 11	Sybase SQL Server License	1	34900.00	34900.00	6000.00	30000.00
	Open Client for Unix	1	795.00	795.00		
	Sybase Software Subotal:			35695.00		30000.00
Client Software						
Tuxedo Pricing from BEA Systems, Inc.						
	Tuxedo 4.2.1 License	1	7500.00	7500.00	1125.00	5625.00
	Tuxedo Software Subtotal:			7500.00		5625.00
	ALL SOFTWARE SUBTOTALS:			**43990.00**		**36765.00**
Discount						
Sun Prepaid Maintenance Discount (Hardware)		15%				**-3612.60**
	TOTAL HARDWARE & SOFTWARE COST (5YR)					**218544.90**
	PRICE PER TPMC					**164.01**

Sun hardware has 1-year warranty bundled.
During the warranty period, maintenance is discounted 50%.
Maintenance prices for Sybase and Tuxedo are yearly rates.

Sun Microsystems	Ultra Enterprise 150	TPC-C Rev 3.2
Sybase, Inc.	C/S w/1 Front-End	Report Date: 11/18/1996

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
$218,544.90	1332.50 tpmC	$164 per tpmC	June 30, 1996

Processors	Database Manager	Operating System	Other Software	Number of Users
1 UltraSPARC 167 MHz	Sybase SQL Server 11.0 RDBMS	Solaris 2.5	Tuxedo Ver. 4.2.1	1160



System Components	Server System	Front End System
Database Nodes:	1 Ultra ENterprise 150	UltraServer 1 Model 140
Processors	1x167 MHz UltraSPARC RISC	1x143 MHz UltraSPARC
Cache memory	32KB (D+I), 512KB external	32KB (D+I), 512KB external
Main memory	512 MB	512 MB
Disk controllers	2* FC25/S + 1 SCSI-2	1xSCSI-2
Disk Drives	56 x 2.1 GB SCSI	1x2.1GB SCSI-2
Total Disk Storage	117.6 GB	2.1 GB
Terminals	1 Console	1 Console
10 BaseT Hubs	None	54 x 24-Port

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark C™ test conducted on the Ultra Enterprise 150 system, running Sybase SQL Server 11.0 RDBMS and Independence Technologies Inc. /iTi Tuxedo 4.2.1.

TPC Benchmark C Metrics

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

Executive Summary Statements

Pages v-vii contain the executive summary of the benchmark result for the Sun Microsystems Ultra Enterprise 150.

lease
cycle

TPC Benchmark™ C Full Disclosure Report Using Sun Microsystems Ultra Enterprise 150 and Sybase SQL Server 11.0 RDBMS

Sun

Sun Microsystems Computer Company
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Revision 1,April 1996
Submitted for review
Compliant with Revision 3.0 of the TPC-C specification