```
/**************************************************************************
 *
 * Filename:        TPCtxn.h
 * Created On:      12.05.1997
 * Created By:      Bettina Kemme
 * Last Modified On:
 * Last Modified By:
 *
 **************************************************************************
 * Contents:  defines, global variables
 **************************************************************************/
#include<stdio.h>

#define CLK_TCK _sysconf(3)     /* 3 is _SC_CLK_TCK */

#define CUST_PER_DIST 1500        /* Customer per District */
#define MAXITEMS 5000             /* Number of Items */
#define NUMBER_WAREHOUSES 1       /* Number of Warehouses */
#define DIST_PER_WARE 10          /* Districts per Warehouse */

#define resize(A) A.arr[A.len]=(char)0
#define TRUE 1
#define FALSE 0

typedef enum Txn_Type {               /* Transaction Type */
  New_Order,
  Payment,
  Order_Status,
  Delivery,
  Stock_Level
} Txn_Type;

void __cg92_used( void );
int NURand(int A, int x, int y);       /* non uniform random number generation */
long RandomNumber(long min, long max); /* uniform random number generation */
void Lastname();                       /* generation of last name of customer */
int neword(long wh);                   /* new order txn */
int payment(long wh);                  /* payment txn */
int ostat(long wh);                    /* order status txn */
int delivery(long wh);                 /* delivery txn */
int slev(long wh, long d);             /* stock level txn */
int ora_startTiming();                 /* statistics */
int ora_stopTiming();                  /* " */
int ora_printStatistics ();            /* " */


void Record(Txn_Type txn_type,double value);  /* Statistics */
double Rand_Exp(int mean);            /* exponential random number generation */
void Record_Init();                           /* Statistics */
void Record_Statistics();                     /* " */

extern int option_debug;    /* 1 if generating debug output */
extern int print_out;       /* if printout per txn generated */
extern FILE *f_plot;        /* statistic and debug output, per terminal */
```

```
/**************************************************************************
 *
 * Filename:         TPCmain.pc
 * Created On:       12.05.1997
 * Created By:       Bettina Kemme
 * Last Modified On:
 * Last Modified By:
 *                                                                        *
 **************************************************************************
 * Contents: emulating a terminal that requests oracle
 **************************************************************************/


#include "TPCtxn.h"
#include <sys/times.h>


void __cg92_used( void )
{/* used for linking gcc compiled code with Oracle Embedded SQL */}

int option_debug = 0;                   /* 1 if generating debug output */
int print_out = 0;                      /* 1 if printout per txn generated */
int option_analysis = 0;                /* 1 if statistic analysis */

/* statistics */
char       ora_statNames[256][64];     /* description of statistics */
float      ora_elapsedTime;            /* total time of execution */

/* time measurements */
struct tms startRes, stopRes;
clock_t    startTim, stopTim;
clock_t    startEmu, stopEmu;
struct tms startEmuRes, stopEmuRes;
int        running_Time;

/* statistic and debug output, per terminal */
FILE *f_plot;


EXEC SQL BEGIN DECLARE SECTION;
VARCHAR userid[20];
VARCHAR password[20];

VARCHAR statName[64];               /* name of specific inf. given by oracle */
int statistics1[256];               /* value of statistic before execution */
int statistics2[256];               /* value of statistic after execution */
int pid, sid;                       /* process id and session id */
int ora_numberOfStatistics;         /* number of statistics given */
int ora_statistics[256];            /* value of statistic for spec. execution */
EXEC SQL END DECLARE SECTION;

void start_terminal(int w_number, int t_id);

EXEC SQL INCLUDE sqlca.h;

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

  int wait_time;
  int number_terminals;

  if (argc<7)
    {
      printf("USAGE: %s <account> <password> <running time> <number_terminals> <print_o
ut> <analysis>\n", argv[0]);
      return;
```



```
    };

  strcpy(userid.arr, argv[1]);
  strcpy(password.arr, argv[2]);
  userid.len=strlen(userid.arr);
  password.len=strlen(password.arr);

  running_Time = atoi(argv[3]);
  number_terminals = atoi(argv[4]);

  /*srandom(atoi(argv[4]));*/          /* Initialize random number stream */

  print_out = atoi(argv[5]);    /* print results of txns to f_plot */

  option_analysis = atoi(argv[6]); /* prints statistic of txns to f_plot */

  if (argc == 8)
    option_debug = atoi(argv[7]);


  Record_Init();                /* for statistics */

  startEmu = times(&startEmuRes); /* get start time of program execution */

  printf("start terminals\n");

  /* start a child process for each terminal */
  for (i=0;i<NUMBER_WAREHOUSES;i++)
    for (j=0;j<number_terminals;j++)
      start_terminal(i+1,j);
}

/* ----------------------------------------------------------------------------
   Terminal Emulation
   Input: warehouse id, terminal id
   ----------------------------------------------------------------------------
   */

void start_terminal(int wh, int terminal_id)
{
  int txn;
  Txn_Type txn_type;
  int wait_time;
  int i,j;

  int which_process;

  char  str[20];

  /* an output file for each terminal */
  sprintf (str, "WH%dTerminal%d",wh, terminal_id);
  f_plot = fopen (str, "w");

  /* each terminal is a single process */
  which_process = fork();

  if (which_process != 0)
    return;
  else
    {
      srandom(terminal_id);     /* Initialize random number stream */

      EXEC SQL WHENEVER SQLERROR GOTO errexit;

      EXEC SQL CONNECT :userid IDENTIFIED BY :password;
      if (option_debug)
        printf("\nTerminal %d of Warehous %d connected to data base\n", terminal_id, wh
);
```

```
      /* start txn requests */
      for (;;)
        {
          stopEmu = times(&stopEmuRes); /* get start time of txn execution */

          /* test ready ? */
          if (((stopEmu-startEmu)/(float)CLK_TCK) > (float) running_Time)
            {
              printf("wh %d, terminal_id %d stops after %f\n",
                     wh, terminal_id,(stopEmu-startEmu)/(float)CLK_TCK);
              break;
            }

          /* choose next transaction */
          txn = RandomNumber(1L,100L);
          if (option_analysis)
            ora_startTiming();
          else
            startTim = times(&startRes); /* get start time of program ex. */
          if (txn <= 43)
            {
              txn_type = New_Order;
              neword(wh);
            }
          else if (txn <= 86)
            {
              txn_type = Payment;
              payment(wh);
            }
          else if (txn <= 91)
            {
              txn_type = Order_Status;
              ostat(wh);
            }
          else if (txn <=  96)
            {
              txn_type = Delivery;
              delivery(wh);
            }
          else
            {
              txn_type = Stock_Level;
              for (j=1;j<=10;j++)
                slev(wh,j);
            }
          if (option_analysis)
            {
              ora_stopTiming();
              ora_printStatistics ();
            }
          else
            {
              stopTim = times(&stopRes); /* get end time of program ex. */
              ora_elapsedTime=(float)(stopTim-startTim) / (float)CLK_TCK;
            }
          Record(txn_type,ora_elapsedTime);
          /* calculate thinking time and keying time between two txn */
          /* we do not have report-input/output */
          switch (txn_type)
            {
            case New_Order:
              wait_time = Rand_Exp(12);
              wait_time += 18;
              break;
            case Payment:
              wait_time = Rand_Exp(12);
              wait_time += 3;
```



```
              break;
            case Order_Status:
              wait_time = Rand_Exp(10);
              wait_time += 2;
              break;
            case Delivery:
            case Stock_Level:
              wait_time = Rand_Exp(5);
              wait_time += 2;
              break;
            }
          /* should wait in real testing */
          /*sleep(wait_time);*/
        }

      EXEC SQL COMMIT WORK RELEASE;
      Record_Statistics();
      exit(0);

    errexit:
      errrpt();
      EXEC SQL WHENEVER SQLERROR CONTINUE;
      printf("\nSQL-ERROR  -->  ROLLBACK\n");
      EXEC SQL ROLLBACK WORK RELEASE;
      exit(0);
    }
}

/* ---------------------------------------------------------------------------------
   Statistic Programs
   ---------------------------------------------------------------------------------
   */

int ora_startTiming() {

  int i;

  EXEC SQL WHENEVER SQLERROR CONTINUE;
  EXEC SQL WHENEVER NOT FOUND CONTINUE;

  /* statistics only gathered when sql_trace == true */
  EXEC SQL ALTER SESSION SET SQL_TRACE TRUE;

  /* V$STATNAME contains the description of the statistics */
  EXEC SQL DECLARE SNAME CURSOR FOR
    SELECT NAME
      FROM V$STATNAME
        ORDER BY STATISTIC#;

  EXEC SQL OPEN SNAME;

  pid=(int)getpid();

  /* V$SESSION contains the sessions ids for all processes */
  EXEC SQL SELECT SID INTO :sid
    FROM   V$SESSION
      WHERE  PROCESS=TO_CHAR(:pid);

  /* V$SESSTAT contains statistics for all sessions */
  EXEC SQL SELECT COUNT(*) INTO :ora_numberOfStatistics
    FROM   V$SESSTAT
      WHERE  SID=:sid;

  /* get description of statistics */
  for (i=0;i<ora_numberOfStatistics;i++)
    {
      EXEC SQL FETCH SNAME INTO :statName;
      resize(statName);
```

```
      strcpy(ora_statNames[i],statName.arr);
    }

  /* get initial values of statistics */
  EXEC SQL SELECT VALUE INTO :statistics1
    FROM   V$SESSTAT
      WHERE  SID=:sid
        ORDER  BY STATISTIC#;


  EXEC SQL CLOSE SNAME;

  startTim = times(&startRes); /* get start time of program execution */
  return 0;
}

int ora_stopTiming() {

  int i;

  stopTim = times(&stopRes); /* get end time of program execution */
  ora_elapsedTime=(float)(stopTim-startTim) / (float)CLK_TCK;

  EXEC SQL WHENEVER SQLERROR CONTINUE;

  EXEC SQL SELECT VALUE INTO :statistics2 /* get end values of statistics */
    FROM   V$SESSTAT
      WHERE  SID=:sid
        ORDER  BY STATISTIC#;

  for(i=0; i<ora_numberOfStatistics; i++) /* calculate stat. for progr. ex. */
    ora_statistics[i]=statistics2[i]-statistics1[i];

  return 0;
}

int ora_printStatistics ()
{
  int i;

  printf("\n S T A T I S T I C\n");

  printf("\nElapsed Time: %f\n\n",ora_elapsedTime);
  /*
     printf("%d %6.6d %s\n",4,ora_statistics[4],ora_statNames[4]);
     printf("%d %6.6d %s\n",5,ora_statistics[5],ora_statNames[5]);
     printf("%d %6.6d %s\n",6,ora_statistics[6],ora_statNames[6]);
     printf("%d %6.6d %s\n",9,ora_statistics[9],ora_statNames[9]);
     printf("%d %6.6d %s\n",11,ora_statistics[6],ora_statNames[11]);
     printf("%d %6.6d %s\n",12,ora_statistics[9],ora_statNames[12]);
     printf("%d %6.6d %s\n",37,ora_statistics[37],ora_statNames[37]);
     printf("%d %6.6d %s\n",38,ora_statistics[38],ora_statNames[38]);
     printf("%d %6.6d %s\n",39,ora_statistics[39],ora_statNames[39]);
     printf("%d %6.6d %s\n",40,ora_statistics[40],ora_statNames[40]);
     printf("%d %6.6d %s\n",39,ora_statistics[43],ora_statNames[43]);

     for (i=76;i<89; i++)
     {
     printf("%d %6.6d %s\n",i,ora_statistics[i],ora_statNames[i]);
     }

     for (i=109;i<117; i++)
     {
     printf("%d %6.6d %s\n",i,ora_statistics[i],ora_statNames[i]);
     }

     printf("%d %6.6d %s\n",130,ora_statistics[130],ora_statNames[130]);
     printf("%d %6.6d %s\n",131,ora_statistics[131],ora_statNames[131]);
```



```
     printf("%d %6.6d %s\n",132,ora_statistics[132],ora_statNames[132]);

     */
  for (i=0;i<ora_numberOfStatistics; i++)
    {
     printf("%d %6.6d %s\n",i,ora_statistics[i],ora_statNames[i]);
    }

  return 0;
}
```

```
/****************************************************************************
 *
 * Filename:         dbs.cpp
 * Created On:       10.05.1997
 * Created By:       Bettina Kemme
 * Last Modified On:
 * Last Modified By:
 *                                                                          *
 ****************************************************************************
 * Contents: Reporting for TPC-C, using CSIM features for that purpose
 ****************************************************************************/

#include <csim.h>
#include "TPCtxn.h"

HIST New_Order_T;
HIST Payment_T;
HIST Order_Status_T;
HIST Delivery_T;
HIST Stock_Level_T;

/* needed because using CSIM */
void sim()
{};


void Record_Init()
{
  New_Order_T = histogram("New_Order_T",10,0.0,10.0);
  Payment_T = histogram("Payment_T",10,0.0,10.0);
  Order_Status_T = histogram("Order_Status_T",10,0.0,10.0);
  Delivery_T = histogram("Delivery_T",10,0.0,50.0);
  Stock_Level_T = histogram("Stock_Level_T",10,0.0,40.0);

  sim();
}


void Record_Statistics()
{
  report();
}

/* record one statistic point */
void Record(Txn_Type txn_type, double value)
{
  switch(txn_type)
    {
    case New_Order:
      record(value,New_Order_T);
      break;
    case Payment:
      record(value,Payment_T);
      break;
    case Order_Status:
      record(value,Order_Status_T);
      break;
    case Delivery:
      record(value,Delivery_T);
      break;
    case Stock_Level:
      record(value,Stock_Level_T);
      break;
    }
}

/* create a random exponential number */
double Rand_Exp(int mean)
```



```
{
  return (expntl(mean));
}
```

```
/******************************************************************************
 *
 * Filename:         TPCtxn.c
 * Created On:       13.05.1997
 * Created By:       Bettina Kemme
 * Last Modified On:
 * Last Modified By:
 *
 ******************************************************************************
 * Contents:  implements the 5 TPC-C benckmark transactions
 ******************************************************************************/

#include "TPCtxn.h"

EXEC SQL INCLUDE sqlca.h;

/* --------------------------------------------------------------------------
   New Order Transaction
   Input: Terminal Warehouse
   Output: Error-Code
   --------------------------------------------------------------------------
   */
int neword(long input_w)
{
  EXEC SQL BEGIN DECLARE SECTION;
  VARCHAR timestamp[20];

  /* warehouse variables */
  long w_id;
  float w_tax;

  /* district variables */
  long d_w_id;
  long d_id;
  float d_tax;
  long d_next_o_id;

  /* customer variables */
  long c_id;
  long c_d_id;
  long c_w_id;
  VARCHAR c_last[16];
  VARCHAR c_credit[2];
  float c_discount;

  /* order variables */
  long o_id;
  long o_c_id;
  long o_d_id;
  long o_w_id;
  long o_carrier_id;
  long o_ol_cnt;
  short o_all_local;
  long ol_number;
  long ol;
  long ol_i_id;
  long ol_supply_w_id;
  long ol_quantity;
  long ol_remote;
  float ol_amount;
  VARCHAR ol_dist_info[24];

  /* item variables */
  long i_id;
  VARCHAR i_name[24];
  float i_price;
  VARCHAR i_data[50];
```



```
  /* stock variables */
  long s_i_id;
  long s_w_id;
  long s_quantity;
  long s_ytd;
  VARCHAR s_dist_01[24];
  VARCHAR s_dist_02[24];
  VARCHAR s_dist_03[24];
  VARCHAR s_dist_04[24];
  VARCHAR s_dist_05[24];
  VARCHAR s_dist_06[24];
  VARCHAR s_dist_07[24];
  VARCHAR s_dist_08[24];
  VARCHAR s_dist_09[24];
  VARCHAR s_dist_10[24];
  VARCHAR s_data[50];

  EXEC SQL END DECLARE SECTION;

  int i;

  int rbk;                          /* abort */
  /* output variables */
  long supware[15];
  long itemid[15];
  long qty[15];
  long s_qty[15];
  long ol_all_remote[15];
  float price[15];
  char iname[24][15];
  float amt[15];
  float total;
  char bg[15];

  int remote;

  w_id = input_w;

  /* random numbers for input variables */
  d_id = RandomNumber(1L,10L);
  d_w_id = w_id;

  c_id = NURand(CUST_PER_DIST/2-15,1, CUST_PER_DIST);
  c_w_id = w_id;
  c_d_id = d_id;

  o_ol_cnt = RandomNumber(5L,15L);
  o_all_local = 1;

  if (option_debug)
    printf ("did %d, c_id %d, ol_cnt %d\n",d_id,c_id,o_ol_cnt);

  rbk = RandomNumber(1L,100L);

  /* select items for the orders */
  for (i=0;i<o_ol_cnt;i++)
    {
      if ((i == o_ol_cnt-1) && (rbk == 1))
        itemid[i] = MAXITEMS+1;
      else
        itemid[i] = NURand(MAXITEMS/2-100,1,MAXITEMS);

      /* item from remote warehouse or not */
      remote = RandomNumber(1L,100L);
      if ((remote > 1) || (NUMBER_WAREHOUSES == 1))
        {
          supware[i] = w_id;
          ol_all_remote[i]= 0;
```

```
      }
    else
      {
        do {
             o_all_local = 0;
             ol_all_remote[i] = 1;
             supware[i] = RandomNumber(1L,NUMBER_WAREHOUSES);
           }
        while (supware[i] == w_id);
      }
    /* amount per item */
    qty[i] = RandomNumber(1L,10L);
  }

strcpy(timestamp.arr,"19970509.00");
timestamp.len = strlen(timestamp.arr);

EXEC SQL WHENEVER NOT FOUND GOTO sqlerr;
EXEC SQL WHENEVER SQLERROR GOTO sqlerr;


if (option_debug)
  printf("start new order\n");

/* customer and warehouse information */
EXEC SQL SELECT c_discount, c_last, c_credit, w_tax
  INTO :c_discount, :c_last, :c_credit, :w_tax
    FROM tpcc.customer, tpcc.warehouse
      WHERE w_id = :w_id AND c_w_id = w_id AND c_d_id = :d_id
        AND c_id = :c_id;

/* get district information */
EXEC SQL SELECT d_next_o_id, d_tax INTO :d_next_o_id, :d_tax
  FROM tpcc.district
    WHERE d_id = :d_id AND d_w_id = :w_id
      FOR UPDATE;

/* increase next order id */
EXEC SQL UPDATE tpcc.district SET d_next_o_id = :d_next_o_id + 1
  WHERE d_id = :d_id AND d_w_id = :w_id;

if (option_debug) {
                  printf("after update district\n");
                  }

o_id=d_next_o_id;
if (option_debug)
  printf("next od -id %d\n",o_id);

/* create new order */
EXEC SQL INSERT INTO tpcc.ORDER_TABLE (o_id, o_d_id, o_w_id,o_c_id,
                                       o_entry_d, o_ol_cnt, o_all_local)
  VALUES (:o_id, :d_id, :w_id, :c_id, :timestamp, :o_ol_cnt, :o_all_local);

if (option_debug) {
                  printf("after insert order_table\n");
                  }


EXEC SQL INSERT INTO tpcc.NEW_ORDER (no_o_id, no_d_id, no_w_id)
  VALUES (:o_id, :d_id, :w_id);

if (option_debug) {
                  printf("after insert new_order\n");
                  }

/* create record for each requested item */
for (ol_number=1; ol_number<=o_ol_cnt;ol_number++)
```



```
  {
    ol_supply_w_id=supware[ol_number-1];
    ol_i_id=itemid[ol_number-1];
    if (option_debug)
      printf("item to modify %d\n",ol_i_id);
    ol_quantity=qty[ol_number-1];
    ol_remote = ol_all_remote[ol_number-1];

    EXEC SQL WHENEVER NOT FOUND GOTO invaliditem;

    i_name.len = 24;

    /* get item information */
    EXEC SQL SELECT i_price, i_name , i_data
      INTO :i_price, :i_name, :i_data
        FROM tpcc.item
          WHERE i_id = :ol_i_id;

    /* only output */
    price[ol_number-1] = i_price;
    resize(i_name);
    strcpy(iname[ol_number-1],i_name.arr);

    EXEC SQL WHENEVER NOT FOUND GOTO sqlerr;

    /* get stock information */
    EXEC SQL SELECT s_quantity, s_data, s_dist_01, s_dist_02,
    s_dist_03, s_dist_04, s_dist_05,
    s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10
      INTO :s_quantity, :s_data, :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
        FROM tpcc.stock
          WHERE s_i_id = :ol_i_id AND s_w_id = :ol_supply_w_id
            FOR UPDATE;

    ol_dist_info.len = 24;
    switch(d_id)
      {
      case 1:
        resize(s_dist_01);
        strcpy(ol_dist_info.arr,s_dist_01.arr);
        break;
      case 2:
        resize(s_dist_02);
        strcpy(ol_dist_info.arr,s_dist_02.arr);
        break;
      case 3:
        resize(s_dist_03);
        strcpy(ol_dist_info.arr,s_dist_03.arr);
        break;
      case 4:
        resize(s_dist_04);
        strcpy(ol_dist_info.arr,s_dist_04.arr);
        break;
      case 5:
        resize(s_dist_05);
        strcpy(ol_dist_info.arr,s_dist_05.arr);
        break;
      case 6:
        resize(s_dist_06);
        strcpy(ol_dist_info.arr,s_dist_06.arr);
        break;
      case 7:
        resize(s_dist_07);
        strcpy(ol_dist_info.arr,s_dist_01.arr);
        break;
      case 8:
```

```
          resize(s_dist_08);
          strcpy(ol_dist_info.arr,s_dist_08.arr);
          break;
        case 9:
          resize(s_dist_09);
          strcpy(ol_dist_info.arr,s_dist_09.arr);
          break;
        case 10:
          resize(s_dist_10);
          strcpy(ol_dist_info.arr,s_dist_10.arr);
          break;
        }

      /* only for output */
      if ( strstr(i_data,"original") && strstr(s_data,"original") )
        bg[ol_number-1] = 'B';
      else
        bg[ol_number-1] = 'G';


      if (s_quantity > (ol_quantity+10))
        s_quantity = s_quantity - ol_quantity;
      else
        s_quantity = s_quantity - ol_quantity + 91;

      s_qty[ol_number-1] = s_quantity;

      EXEC SQL UPDATE tpcc.stock
        SET s_quantity = :s_quantity,
        s_ytd = s_ytd + :ol_quantity,
        s_order_cnt = s_order_cnt + 1,
        s_remote_cnt = s_remote_cnt + :ol_remote
          WHERE s_i_id = :ol_i_id AND s_w_id = :ol_supply_w_id;

      if (option_debug) {
                          printf("after udpate stock\n");
                        }

      ol_amount = ol_quantity * i_price * (1+w_tax+d_tax) * (1-c_discount);

      /* only for output */
      amt[ol_number-1]=ol_amount;
      total += ol_amount;


      EXEC SQL INSERT INTO tpcc.order_line (ol_o_id, ol_d_id, ol_w_id,
                                            ol_number, ol_i_id, ol_supply_w_id,
                                            ol_quantity, ol_amount, ol_dist_info)
        VALUES (:o_id, :d_id, :w_id, :ol_number, :ol_i_id,
                :ol_supply_w_id, :ol_quantity, :ol_amount, :ol_dist_info);

      if (option_debug) {
                          printf("after inser Order_line\n");
                        }
    }

  if (option_debug)
    printf("ready new order\n");

  if (print_out)
    {
      fprintf(f_plot,"\n\nNew Order Transaction:\n");
      resize(c_last),resize(c_credit);
      fprintf(f_plot,"WH-ID: %d with Tax %2.3f, Distr. ID: %d with Tax %2.3f\n",
              w_id,w_tax,d_id,d_tax);
      fprintf(f_plot,"Customer ID: %d, Name %s, Credit %s, Discount %2.f\n",
              c_id,c_last.arr,c_credit.arr,c_discount);
      fprintf(f_plot,"Order ID: %d, Number Orders %d, Date %s\n",
```



```
              o_id,o_ol_cnt,timestamp.arr);
      fprintf(f_plot,"Single Orders:\n");
      for (i=0;i<o_ol_cnt;i++)
        {
          fprintf(f_plot,"WH: %d, Item ID: %d, Quantity: %d, Rem. Quant.: %d, bg: %c, p
rice %2.2f, amount %2.3f\n",
                  supware[i],itemid[i],qty[i],s_qty[i],bg[i],price[i],amt[i]);
        }
    }


  /*End Order Lines*/
  EXEC SQL COMMIT WORK;
  return(0);

 invaliditem:
  EXEC SQL ROLLBACK WORK;
  printf("Item number is not valid");
  return(0);

 sqlerr:
  errrpt();
  return(1);
}

/* ----------------------------------------------------------------------------
   Payment Transaction
   Input: Terminal Warehouse
   Output: Error-Code
   ----------------------------------------------------------------------------
   */
int payment(long input_w)
{
  EXEC SQL BEGIN DECLARE SECTION;
  VARCHAR timestamp[20];

  /* warehouse */
  long w_id;
  VARCHAR w_name[10];
  VARCHAR w_street_1[20];
  VARCHAR w_street_2[20];
  VARCHAR w_city[20];
  VARCHAR w_state[2];
  VARCHAR w_zip[9];

  /* district */
  long d_w_id;
  long d_id;
  VARCHAR d_name[10];
  VARCHAR d_street_1[20];
  VARCHAR d_street_2[20];
  VARCHAR d_city[20];
  VARCHAR d_state[2];
  VARCHAR d_zip[9];

  /* customer */
  long c_id;
  long c_d_id;
  long c_w_id;
  VARCHAR c_first[16];
  VARCHAR c_middle[2];
  VARCHAR c_last[16];
  VARCHAR c_street_1[20];
  VARCHAR c_street_2[20];
  VARCHAR c_city[20];
  VARCHAR c_state[2];
  VARCHAR c_zip[9];
  VARCHAR c_phone[16];
```

```
  VARCHAR c_since[11];
  VARCHAR c_credit[2];
  long c_credit_lim;
  float c_discount;
  float c_balance;
  VARCHAR c_data[500];
  VARCHAR c_new_data[500];

  /* history */
  VARCHAR h_data[24];
  float   h_amount;

  EXEC SQL END DECLARE SECTION;

  int i;
  int namecnt;
  int remote;
  int byname;

  /* generation of input */
  w_id = input_w;
  d_id = RandomNumber(1L,10L);
  c_d_id = d_id;

  if (option_debug)
    printf("start new payment\n");

  strcpy(timestamp.arr,"19970509.00");
  timestamp.len = strlen(timestamp.arr);

  if (NUMBER_WAREHOUSES > 1)
    remote = RandomNumber(1L,100L);
  else
    remote = 1;

  /* customer pays at remote or own warehouse */
  if (remote <= 85)
    {
      c_w_id = w_id;
    }
  else
    {
      do {
           c_w_id = RandomNumber(1L,NUMBER_WAREHOUSES);
         } while (c_w_id == w_id);
    }

  /* customer selected by name or by id */
  byname = RandomNumber(1L,100L);
  if (byname <= 60)
    {
      Lastname(NURand(255,0,999),&c_last);
      byname = TRUE;
    }
  else
    {
      c_id = NURand(CUST_PER_DIST/2-15,1, CUST_PER_DIST);
      byname = FALSE;
    }
  if (option_debug)
    printf("remote %d, byname %d, c_id %d, c_w_id %d, c_d_id %d\n",
           remote,byname,c_id,c_w_id,c_d_id);

  h_amount = (float) RandomNumber(1L,500000L)/100.00;

  EXEC SQL WHENEVER NOT FOUND GOTO sqlerr;
  EXEC SQL WHENEVER SQLERROR GOTO sqlerr;
```



```

  EXEC SQL UPDATE tpcc.warehouse
    SET w_ytd = w_ytd + :h_amount
      WHERE w_id=:w_id;

  EXEC SQL SELECT w_street_1, w_street_2, w_city, w_state, w_zip, w_name
    INTO :w_street_1, :w_street_2, :w_city, :w_state, :w_zip, :w_name
      FROM tpcc.warehouse
        WHERE w_id=:w_id;

  if (option_debug){
                      printf("selected and updated warehouse\n");
                    }

  EXEC SQL UPDATE tpcc.district
    SET d_ytd = d_ytd + :h_amount
      WHERE d_w_id=:w_id AND d_id=:d_id;

  EXEC SQL SELECT d_street_1, d_street_2, d_city, d_state, d_zip, d_name
    INTO :d_street_1, :d_street_2, :d_city, :d_state, :d_zip, :d_name
      FROM tpcc.district
        WHERE d_w_id=:w_id AND d_id=:d_id;

  if (option_debug){
                      printf("selected and updated district\n");
                    }

  resize(w_name),resize(d_name);
  h_data.arr[0] = '\0';
  strcpy(h_data.arr,w_name.arr);
  strcat(h_data.arr,"    ");
  strcat(h_data.arr,d_name.arr);
  h_data.len = strlen(h_data.arr);

  if (option_debug)
    printf("h_data: %s\n",h_data.arr);


  if (byname)
    {
      /* select customer by name, it is chosen from all customers */
      /* with the same name the one in the middle of all returned customers */
      EXEC SQL SELECT count(c_id)
        INTO :namecnt
          FROM tpcc.customer
            WHERE c_last=:c_last AND c_d_id=:c_d_id AND c_w_id=:c_w_id;

      if (option_debug)
        printf("namecnt %d\n",namecnt);

      EXEC SQL DECLARE c_byname CURSOR FOR
        SELECT c_first, c_middle, c_id, c_street_1, c_street_2, c_city,
        c_state, c_zip, c_phone, c_credit, c_credit_lim, c_discount,
        c_balance, c_since
          FROM tpcc.customer
            WHERE c_w_id=:c_w_id AND c_d_id=:c_d_id AND c_last=:c_last
              ORDER BY c_first;

      if (option_debug){
                          printf("fetch customer by name\n");
                        }

      EXEC SQL OPEN c_byname;
      if (namecnt%2)
        namecnt++;
      namecnt = namecnt/2;
      /*Locate midpoint customer;*/
      for (i=0; i<namecnt; i++)
```

```
        {
          EXEC SQL FETCH c_byname
            INTO :c_first, :c_middle, :c_id, :c_street_1, :c_street_2, :c_city,
            :c_state, :c_zip, :c_phone, :c_credit, :c_credit_lim,
            :c_discount, :c_balance, :c_since;
        }
      EXEC SQL CLOSE c_byname;

      if (option_debug){
                         printf("customer fetched by name\n");
                       }
    }
  else
    {
      /* select customer by id */
      EXEC SQL SELECT 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_since
        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_since
          FROM tpcc.customer
            WHERE c_w_id=:c_w_id AND c_d_id=:c_d_id AND c_id=:c_id;

      if (option_debug){
                         printf("customer fetched by id\n");
                       }
    }
  c_balance += h_amount;
  resize(c_credit);
  if (strstr(c_credit.arr, "BC") )
    {
      EXEC SQL SELECT c_data
        INTO :c_data
          FROM tpcc.customer
            WHERE c_w_id=:c_w_id AND c_d_id=:c_d_id AND c_id=:c_id;

      c_new_data.len =
        sprintf(c_new_data.arr,"| %4d %2d %4d %2d %4d $%7.2f %12c %24c",
                c_id,c_d_id,c_w_id,d_id,w_id,h_amount,timestamp.arr, h_data.arr);
      strncat(c_new_data.arr,c_data.arr,500-c_new_data.len);
      c_new_data.len = strlen(c_new_data.arr);

      EXEC SQL UPDATE tpcc.customer
        SET c_balance = :c_balance, c_ytd_payment = c_ytd_payment + :h_amount,
        c_data = :c_new_data
          WHERE c_w_id = :c_w_id AND c_d_id = :c_d_id AND c_id = :c_id;

      if (option_debug){
                         printf("customer + data updatede\n");
                       }
    }
  else
    {
      EXEC SQL UPDATE tpcc.customer
        SET c_balance = :c_balance, c_ytd_payment = c_ytd_payment + :h_amount
          WHERE c_w_id = :c_w_id AND c_d_id = :c_d_id AND c_id = :c_id;
      if (option_debug){
                         printf("customer updatede\n");
                       }
    }

  EXEC SQL INSERT
    INTO tpcc.history (h_c_d_id, h_c_w_id, h_c_id, h_d_id, h_w_id, h_date,
                       h_amount, h_data)
      VALUES (:c_d_id, :c_w_id, :c_id, :d_id, :w_id, :timestamp, :h_amount,
              :h_data);
```



```
  if (option_debug){
                     printf("history updatede\n");
                   }

  if (print_out)
    {
      fprintf(f_plot,"\n\nPayment Transaction:\n");
      resize(w_street_1),resize(w_street_2),resize(w_city),resize(w_state),
      resize(w_zip);
      resize(d_street_1),resize(d_street_2),resize(d_city),resize(d_state),
      resize(d_zip);
      resize(c_street_1),resize(c_street_2),resize(c_city),resize(c_state),
      resize(c_zip),resize(c_phone);
      resize(c_last),resize(c_first),resize(c_middle),resize(c_credit);
      fprintf(f_plot,"Warehouse ID: %d, Street %s %s, City %s, State %s, ZIP %s\n",
              w_id,w_street_1.arr, w_street_2.arr,w_city.arr,w_state.arr,w_zip.arr);
      fprintf(f_plot,"District ID: %d, Street %s %s, City %s, State %s, ZIP %s\n",
              d_id,d_street_1.arr, d_street_2.arr,d_city.arr,d_state.arr,d_zip.arr);
      fprintf(f_plot,"Customer ID: %d, C-WH %d, C-DS %d, Name %s, First %s, Middle %s\n
",
              c_id, c_w_id, c_d_id, c_last.arr, c_first.arr, c_middle.arr);
      fprintf(f_plot,"Street %s %s, City %s, State %s, ZIP %s Phone %s\n",
              c_street_1.arr, c_street_2.arr,c_city.arr,c_state.arr,c_zip.arr,

              c_phone.arr);
      fprintf(f_plot,"Credit %s, Credit_lim %d, Discount %2.f, Balance %.2f\n",
              c_credit.arr,c_credit_lim,c_discount,c_balance);
      if (strstr(c_credit.arr, "BC") )
        fprintf(f_plot,"Data: %*s\n",200,c_new_data.arr);
    }

  EXEC SQL COMMIT WORK;
  return(0);

 sqlerr:
  errrpt();
  return(1);
}

/* ----------------------------------------------------------------------------
   Order Status Transaction
   Input: Terminal Warehouse
   Output: Error-Code
   ----------------------------------------------------------------------------
   */
int ostat(long input_w)
{
  EXEC SQL BEGIN DECLARE SECTION;

  /* warehouse and district */
  long w_id;
  long d_id;

  /* customer */
  long c_id;
  long c_d_id;
  long c_w_id;
  VARCHAR c_first[16];
  VARCHAR c_middle[2];
  VARCHAR c_last[16];
  float c_balance;

  /* order */
  long o_id;
  long o_c_id;
  long o_d_id;
  long o_w_id;
  long o_carrier_id;
```

```
  VARCHAR o_entry_d[12];

  /* order line */
  long ol_w_id;
  long ol_d_id;
  long ol_o_id;
  long ol_i_id[15];
  long ol_supply_w_id[15];
  long ol_quantity[15];
  float ol_amount[15];

  EXEC SQL END DECLARE SECTION;

  int o_ol_cnt;
  int i,j;
  int namecnt;
  int byname;
  int order_found = FALSE;

  if (option_debug)
    printf("start new order stats\n");

  w_id = input_w;
  d_id = RandomNumber(1L,10L);


  /* choose customer by name or by id */
  byname = RandomNumber(1L,100L);
  if (byname <= 60)
    {
      Lastname(NURand(255,0,999),&c_last);
      byname = TRUE;
    }
  else
    {
      c_id = NURand(CUST_PER_DIST/2-15,1, CUST_PER_DIST);
      byname = FALSE;
    }

  if (option_debug)
    printf("byname %d, c_id %d, w_id %d, d_id %d\n",byname,c_id,w_id,d_id);


  EXEC SQL WHENEVER NOT FOUND GOTO sqlerr;
  EXEC SQL WHENEVER SQLERROR GOTO sqlerr;
  if (byname)
    {
      /* customer by name */
      /* select the one in the middle of the list of all customers */
      /* with the generated name */
      EXEC SQL SELECT count(c_id)
        INTO :namecnt
          FROM tpcc.customer
            WHERE c_last=:c_last AND c_d_id=:d_id AND c_w_id=:w_id;

      if (option_debug)
        printf("namecnt %d\n",namecnt);

      EXEC SQL DECLARE c_name CURSOR FOR
        SELECT c_balance, c_first, c_middle, c_id
          FROM tpcc.customer
            WHERE c_last=:c_last AND c_d_id=:d_id AND c_w_id=:w_id
              ORDER BY c_first;

      EXEC SQL OPEN c_name;
      if (namecnt%2)
        namecnt++;
      namecnt = namecnt / 2;
```



```
      /* Locate midpoint customer*/
      for (i=0; i<namecnt; i++)
        {
          EXEC SQL FETCH c_name
            INTO :c_balance, :c_first, :c_middle, :c_id;
        }
      EXEC SQL CLOSE c_name;
      if (option_debug)
        printf("c_id %d, w_id %d, d_id %d\n",c_id,w_id,d_id);

    }
  else
    {
      /* customer by id */

      EXEC SQL SELECT c_balance, c_first, c_middle, c_last
        INTO :c_balance, :c_first, :c_middle, :c_last
          FROM tpcc.customer
            WHERE c_id=:c_id AND c_d_id=:d_id AND c_w_id=:w_id;
    }

  /* find last order of the customer */
  EXEC SQL DECLARE o_cursor CURSOR FOR
    SELECT o_id, nvl(o_carrier_id,-1), o_entry_d
      FROM tpcc.order_table
        WHERE o_w_id = :w_id AND o_d_id = :d_id AND o_c_id = :c_id
          ORDER BY o_id DESC;

  EXEC SQL WHENEVER NOT FOUND GOTO noorder;

  EXEC SQL OPEN o_cursor;
  EXEC SQL FETCH o_cursor
    INTO :o_id, :o_carrier_id, :o_entry_d;

  resize(o_entry_d);
  order_found = TRUE;

  if (option_debug)
    printf("order_found %d c_id %d o_id %d, o_carrier_id %d, o_entry_d %s\n",
           order_found, c_id, o_id,o_carrier_id,o_entry_d.arr);

 noorder:
  EXEC SQL CLOSE o_cursor;

  /* find order lines to the order */
  if (order_found)
    {
      EXEC SQL DECLARE c_line
        CURSOR FOR SELECT ol_i_id, ol_supply_w_id, ol_quantity, ol_amount
          FROM tpcc.order_line
            WHERE ol_o_id=:o_id AND ol_d_id=:d_id AND ol_w_id=:w_id;

      EXEC SQL OPEN c_line;
      EXEC SQL WHENEVER NOT FOUND GOTO endloop;

      o_ol_cnt=0;
      while (TRUE)
        {
          EXEC SQL FETCH c_line
            INTO :ol_i_id[o_ol_cnt], :ol_supply_w_id[o_ol_cnt],
            :ol_quantity[o_ol_cnt], :ol_amount[o_ol_cnt];
          o_ol_cnt++;
        }

    endloop:
      EXEC SQL CLOSE c_line;
    }
```

```
  if (print_out)
    {
      fprintf(f_plot,"\n\nStatus Order Transaction:\n");
      resize(c_last),resize(c_first),resize(c_middle);
      fprintf(f_plot,"Warehouse ID: %d, District ID %d\n",w_id,d_id);
      fprintf(f_plot,"Customer ID: %d, Name %s, First %s, Middle %s, Balance %.3f\n",
              c_id, c_last.arr, c_first.arr, c_middle.arr, c_balance);
      if (order_found)
        {
          resize(o_entry_d);
          fprintf(f_plot,"Order ID: %d, Date %s, Carrier %d\n",
                  o_id,o_entry_d.arr,o_carrier_id);
          fprintf(f_plot,"Single Orders:\n");
          for (j=0;j<o_ol_cnt;j++)
            {
              fprintf(f_plot,"WH: %d, Item ID: %d, Quantity: %d, amount %.3f\n",
                      ol_supply_w_id[j],ol_i_id[j],ol_quantity[j],ol_amount[j]);
            }
        }
      else
        printf("no orders for customer\n");
    }


  EXEC SQL COMMIT WORK;
  return(0);

 sqlerr:
  errrpt();
  return(0);
}

/* ---------------------------------------------------------------------------
   Delivery Transaction
   Input: Terminal Warehouse
   Output: Error-Code
   ---------------------------------------------------------------------------
   */
int delivery(long input_w)
{
  EXEC SQL BEGIN DECLARE SECTION;
  VARCHAR timestamp[20];

  /* warehouse, district, customer */
  long w_id;
  long d_id;
  long c_id;

  /* order, order_line , new-order */
  long o_carrier_id;
  VARCHAR ol_delivery_d[12];
  long ol_total;

  long no_o_id;
  EXEC SQL END DECLARE SECTION;

  o_carrier_id = RandomNumber(1L,10L);
  strcpy(timestamp.arr,"19970509.00");
  timestamp.len = strlen(timestamp.arr);

  EXEC SQL WHENEVER SQLERROR GOTO sqlerr;

  w_id = input_w;

  /* For each district in warehouse */
  if (option_debug)
    printf("Delivery W: %d\n", w_id);
  for (d_id=1; d_id<=DIST_PER_WARE; d_id++)
```



```
    {
      EXEC SQL WHENEVER NOT FOUND GOTO sqlerr;

      /* finde oldest not delivered order of an district */
      EXEC SQL DECLARE c_no CURSOR FOR
        SELECT no_o_id
          FROM tpcc.new_order
            WHERE no_d_id = :d_id AND no_w_id = :w_id
              ORDER BY no_o_id ASC
                FOR UPDATE;

      EXEC SQL OPEN c_no;

      EXEC SQL WHENEVER NOT FOUND continue;

      EXEC SQL FETCH c_no
        INTO :no_o_id;

      /* remove record from the new_order table */
      EXEC SQL DELETE FROM tpcc.new_order
        WHERE CURRENT OF c_no;
      EXEC SQL CLOSE c_no;

      /* update order information */
      EXEC SQL SELECT o_c_id
        INTO :c_id
          FROM tpcc.order_table
            WHERE o_id = :no_o_id AND o_d_id = :d_id AND o_w_id = :w_id
              FOR UPDATE;

      EXEC SQL UPDATE tpcc.order_table
        SET o_carrier_id = :o_carrier_id
          WHERE o_id = :no_o_id AND o_d_id = :d_id AND o_w_id = :w_id;

      EXEC SQL UPDATE tpcc.order_line
        SET ol_delivery_d = :timestamp
          WHERE ol_o_id = :no_o_id AND ol_d_id = :d_id AND ol_w_id = :w_id;

      EXEC SQL SELECT SUM(ol_amount)
        INTO :ol_total
          FROM tpcc.order_line
            WHERE ol_o_id = :no_o_id AND ol_d_id = :d_id AND ol_w_id = :w_id;

      /* update customer balance */
      EXEC SQL UPDATE tpcc.customer
        SET c_balance = c_balance + :ol_total,
        c_delivery_cnt = c_delivery_cnt + 1
          WHERE c_id = :c_id AND c_d_id = :d_id AND c_w_id = :w_id;

      EXEC SQL COMMIT WORK;
      if (print_out)
        fprintf(f_plot,"Delivery for district %d done: order %d\n",
                d_id, no_o_id);

    } /* end for */

  if (print_out)
    fprintf(f_plot,"Delivery done\n");

  EXEC SQL COMMIT WORK;
  return(0);

 sqlerr:
  if (print_out)
    fprintf(f_plot,"Delivery error done\n");
  errrpt();
  return(0);
}
```

```
/* -----------------------------------------------------------------------------
   Stock-Level Transaction
   Input: Terminal Warehouse, District
   Output: Error-Code
   -----------------------------------------------------------------------------
   */
int slev(long input_w, long input_d)
{
  EXEC SQL BEGIN DECLARE SECTION;

  long w_id;
  long d_id;
  long o_id;
  long stock_count;
  EXEC SQL END DECLARE SECTION;

  int threshold;

  w_id = input_w;
  d_id = input_d;

  threshold = RandomNumber(10L,20L);

  EXEC SQL WHENEVER NOT FOUND GOTO sqlerr;
  EXEC SQL WHENEVER SQLERROR GOTO sqlerr;

  if (option_debug)
    printf("start new stocklevel wh %d district %d\n", w_id,d_id);

  EXEC SQL SELECT d_next_o_id
    INTO :o_id
      FROM tpcc.district
        WHERE d_w_id=:w_id AND d_id=:d_id;

  /* look for the items that were ordered in the last 20 orders */
  /* whether their amount on stock is under a certain threshold */
  EXEC SQL SELECT COUNT(DISTINCT (s_i_id))
    INTO :stock_count
      FROM tpcc.order_line, tpcc.stock
        WHERE ol_w_id=:w_id
          AND ol_d_id=:d_id
            AND ol_o_id<:o_id
              AND ol_o_id>=:o_id-20
                AND s_w_id=:w_id
                  AND s_i_id=ol_i_id
                    AND s_quantity < :threshold;

  EXEC SQL COMMIT WORK;
  if (print_out)
    {
      fprintf(f_plot,"\n\nStock Level Transaction:\n");
      fprintf(f_plot,"Warehouse: %d, District %d, Threshold %d, stock_count %d\n",
              w_id,d_id,threshold,stock_count);
    }
  return(0);

 sqlerr:
  errrpt();
  return(0);
}

/* generate a random number between min and max - uniformly distributed */
long RandomNumber(long min, long max)
{
  int r = random();
  if (r < 0)
    r = -r;
```



```
  return(min + (r%(max-min+1)));
}

/* generate a random number between x and y - non uniformly distributed */
/* function provided by tpc-c specification */
int NURand(int A, int x, int y)
{
  int C=25;

  return (((RandomNumber(0,A) | RandomNumber(x,y)) + C) % (y-x+1)) + x;
}

/* generate a last name by the rules of the tpc-c specification */
void Lastname(num, name)
     int num;
     VARCHAR *name;
{
  int i;
  static char *n[] = {"BAR", "OUGHT", "ABLE", "PRI", "PRES", "ESE",
                      "ANTI", "CALLY", "ATION", "EING"};
  strcpy(name->arr,n[num/100]);
  strcat(name->arr,n[(num/10)%10]);
  strcat(name->arr,n[num%10]);
  name->len = strlen(name->arr);
  return;
}


/* -----------------------------------------------------------------------------
   errrpt prints the ORACLE error msg and number
   ----------------------------------------------------------------------------- */

errrpt()
{
  printf("\n%.70s (%d)\n", sqlca.sqlerrm.sqlerrmc, sqlca.sqlcode);
  return(0);
}
```