Contents

1. Introduction .................................................................................................................. 1
   1.1 Overview ............................................................................................................... 1
   1.2 Overview of ODBC ......................................................................................... 1

2. Quick Tour .................................................................................................................. 4
   2.1 Introductory Tutorial ...................................................................................... 4
       Init 6
       Define .................................................................................................................. 7
       Manage .............................................................................................................. 8
       Done ................................................................................................................... 10
       Additional Resources ..................................................................................... 11
   2.2 Relationships ....................................................................................................... 12
       Init 14
       Define ................................................................................................................ 15
       Manage ............................................................................................................ 18
       Done ................................................................................................................ 22
       Additional Resources ..................................................................................... 23
   2.3 Record/Row Locking ............................................................................................ 24
       Init 25
       Define ................................................................................................................ 26
       Manage ............................................................................................................ 27
       Done ................................................................................................................ 30
       Additional Resources ..................................................................................... 31
   2.4 Transaction Processing ......................................................................................... 32
       Init 33
       Define ................................................................................................................ 34
       Manage ............................................................................................................ 37
       Done ................................................................................................................ 42
       Additional Resources ..................................................................................... 43

3. Configuring c-treeACE SQL Data Sources ............................................................... 44
   3.1 Introduction ........................................................................................................... 44
   3.2 Configuring Data Sources with the ODBC Data Source Administrator .......... 44
       Invoking the ODBC Data Source Administrator Utility ......................................... 45
       c-treeACE ODBC Data Sources ...................................................................... 45
       Modifying and Deleting c-treeACE SQL Data Sources ........................................ 47
   3.3 ODBC Data Source and Driver Information FORMAT ....................................... 48
       Windows Registry Entries ............................................................................... 48
3.4 Query Timeout Options .................................................................48
3.5 ODBC Driver Socket SEND/RECV Timeout ................................49
3.6 Driver Login Timeout .................................................................49
3.7 Using Reserved Keywords with Microsoft Excel .........................50

4. ODBC Conformance Notes ............................................................51
4.1 Introduction .................................................................................51
4.2 General ODBC and SQL Support Levels .......................................51
4.3 Responses to SQLGetInfo ..........................................................51
4.4 Supported Environment, Connection, and Statement Attributes ......68
4.5 Supported Functions ....................................................................71
4.6 Supported Data Types ...................................................................74
4.7 Thread Safety of c-treeACE ODBC Driver .....................................75
4.8 Error Messages ............................................................................76

5. Direct Link c-treeACE SQL ODBC Driver ......................................80
5.1 Introduction .................................................................................80
5.2 Linking the c-treeACE SQL Direct Link Driver ..............................80
5.3 Unix/Linux ODBC Managers .......................................................81

6. Migration to c-treeACE SQL ODBC ..............................................83
6.1 Migration From c-tree Plus ODBC to c-treeACE SQL ODBC ........83

7. Troubleshooting ............................................................................84
7.1 c-treeACE setup error 126 during ODBC installation phase ..........84

8. Glossary .......................................................................................85

9. Index ............................................................................................90
1. Introduction

1.1 Overview

This chapter introduces ODBC, the c-treeACE SQL ODBC driver, and how they work together to provide access from a wealth of desktop tools to c-treeACE SQL environments:

- A c-treeACE SQL environment is an SQL interface over c-tree.
- ODBC: Microsoft’s Open Data Base Connectivity has become a widely accepted database access standard on the desktop.
- Desktop tools and applications that support ODBC can access any data source supported by an ODBC driver. These tools include client/server development, query, and personal productivity tools.

In V11 and later, c-treeACE SQL ODBC drivers for Unix are available on AIX and Solaris (Sparc) platforms.

1.2 Overview of ODBC

The Open Database Connectivity (ODBC) interface from Microsoft has emerged as the standard mechanism for client applications to access data from a variety of different sources through a single interface. Users of applications supporting ODBC merely select a new database from a point-and-click menu to connect transparently to that data source.

To become accessible from ODBC client applications, database environments must provide software, called a driver, on the client system where the application resides. The driver translates the standard ODBC function calls into calls the data source can process, and returns data to the application. Each data source provides a driver on the client system for applications to use to access data from that source.
The c-treeACE SQL ODBC Driver extends this plug-and-play interoperability to c-treeACE SQL. It allows any Microsoft Windows tool or application that supports the ODBC call library to easily use c-treeACE SQL as a data source. With it, applications based on tools such as and Visual Basic can include c-treeACE SQL as a data source.

The ODBC interface specifies two major components:

- A library of function calls that allow applications to connect with a database system and issue statements through an application programming interface (API)
- Syntax for Structured Query Language (SQL) statements, based on existing standards

ODBC drivers fit in as a layer of “middleware” in the ODBC architecture. The ODBC architecture includes the following layers:

**Application**
An ODBC application is any program that calls ODBC functions and uses them to issue SQL statements. For example, many vendors have added ODBC support to their existing Windows-based tools, such as PowerBuilder™ and Impromptu®, so those tools can use ODBC for data access.

**ODBC Driver Manager**
A Microsoft-supplied dynamic-link library (DLL) that routes calls from an application to the appropriate ODBC driver for a data source. An application sees the ODBC driver manager and a driver as a single entity that processes requests to a particular data source. The ODBC driver manager loads the requested driver in response to an application’s call to the ODBC SQLConnect() or SQLDriverConnect() functions.

**ODBC Driver**
A dynamic link library (DLL) that processes ODBC function calls for a specific data source. The driver connects to the data source and translates the standard SQL statements into syntax the data source can process. It also returns any requested data to the application. There are ODBC drivers for every major database system.

**Data Source**
The combination of a database system, the operating system it uses, and any network software required to access it. (ODBC defines a database system (DBMS) as any vendor’s implementation of a data access system that provides an SQL interface.)
The following figure shows the components involved in a typical ODBC environment.

**Figure 1: ODBC Architecture**
2. Quick Tour

2.1 Introductory Tutorial

If you are using c-treeRTG, adjust the path to match your version, e.g., V2.2.0.RTG, and replace sdk with Driver.

This tutorial will take you through the basic use of the c-treeACE SQL ODBC Interface.

Like all other examples in the c-tree tutorial series, this tutorial simplifies the creation and use of a database into four simple steps: Initialize(), Define(), Manage(), and You’re Done()!

**Tutorial #1: Introductory - Simple Single Table**

We wanted to keep this program as simple as possible. This program does the following:

- Initialize() - Connects to the c-treeACE Database Engine.
- Define() - Defines and creates a "customer master" (custmast) table/file.
- Manage() - Adds a few rows/records; Reads the rows/records back from the database; displays the column/field content; and then deletes the rows/records.
- Done() - Disconnects from c-treeACE Database Engine.

Note our simple mainline:

```c
/*
 * main()
 *
 * The main() function implements the concept of "init, define, manage
 * and you're done..."
 */
int main(int argc, char* argv[])
{
```
Initialize();  
Define();  
Manage();  
Done();

printf("\nPress <ENTER> key to exit . . .\n");
getchar();
return(0);
}

We suggest opening the source code with your own editor.

Continue now to review these four steps.
Init

First we need to open a connection to a database by providing the c-treeACE Database Engine with a user name, password and the database name.

Below is the code for Initialize():

```c
/*
 * Initialize()
 *
 * Perform the minimum requirement of logging onto the c-tree Server
 */
void Initialize(void)
{
    RETCODE rc;

    printf("INIT\n");

    /* allocate environment handle */
    if ((rc = SQLAllocEnv(&hEnv)) != SQL_SUCCESS)
        Handle_Error(0, NULL, "SQLAllocEnv()");

    /* allocate connection handle */
    if ((rc = SQLAllocConnect(hEnv, &hDbc)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_ENV, hEnv, "SQLAllocConnect()");

    /* connect to server */
    printf("Logon to server...\n");
    if ((rc = SQLConnect(hDbc, MY_DSN, SQL_NTS, "admin", SQL_NTS, "ADMIN", SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLConnect()");

    /* allocate statement handle */
    if ((rc = SQLAllocHandle(SQL_HANDLE_STMT, hDbc, &hStmt)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLAllocHandle(SQL_HANDLE_STMT)");
}
```
Define

The define step is where specific data definitions are established by your application and/or process. This involves defining columns/fields and creating the tables/files with optional indices.

Below is the code for Define():

```c
/*
 * Define()
 *
 * Create the table for containing a list of existing customers
 */
void Define(void)
{
    RETCODE  rc;

    printf("DEFINE\n");

    /* create table */
    printf("\tcCreate table...\n");
    if ((rc = SQLExecDirect(hStmt,
        "CREATE TABLE custmast ( \
            cm_custnumb CHAR(4), \
            cm_custzipc CHAR(9), \
            cm_custstat CHAR(2), \
            cm_custrtng CHAR(1), \
            cm_custname VARCHAR(47), \
            cm_custaddr VARCHAR(47), \
            cm_custcity VARCHAR(47))",
        SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(CREATE TABLE)");
}
```
Manage

The manage step provides data management functionality for your application and/or process.

Below is the code for Manage():

```c
/*
 * Manage()
 *
 * This function performs simple record functions of add, delete and gets
 */
void Manage(void)
{
    printf("MANAGE\n");
    /* delete any existing records */
    Delete_Records();
    /* populate the table with data */
    Add_Records();
    /* display contents of table */
    Display_Records();
}

/*
 * Delete_Records()
 *
 * This function deletes all the records in the table
 */
void Delete_Records(void)
{
    RETCODE  rc;
    printf("\tDelete records...\n");
    if ((rc = SQLExecDirect(hStmt, "DELETE FROM custmast", SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(DELETE)");
}

/*
 * Add_Records()
 *
 * This function adds records to a table from an array of strings
 */
void Add_Records(void)
{
    RETCODE     rc;
    SQLINTEGER  i;
    SQLTCHAR    sCommand[512];
    SQLTCHAR    *data[] = {
        "('1000','92867','CA','1','Bryan Williams','2999 Regency','Orange')",
```
SQLINTEGER nRecords = sizeof(data) / sizeof(data[0]);

/* add one record at time to table */
for (i = 0; i < nRecords; i++)
{
    strcpy (sCommand, "INSERT INTO custmast VALUES ");
    strcat (sCommand, data[i]);
    if ((rc = SQLExecDirect(hStmt, sCommand ,SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(INSERT)");
}

.IsDBNull records...\n"

/* Display_Records()*/
/* This function displays the contents of a table. */
void Display_Records(void)
{
    RETCODE rc;
    SQLTCHAR custnumb[4+1];
    SQLTCHAR custname[47+1];
    SDWORD cbData;

    printf("\tDisplay records...\n");

    rc = SQLExecDirect(hStmt, "SELECT * FROM custmast" ,SQL_NTS);
    if (rc != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(SELECT)");

    /* fetch and display each individual record */
    while ((rc = SQLFetch(hStmt)) == SQL_SUCCESS)
    {
        SQLGetData(hStmt, 1, SQL_C_CHAR, custnumb, sizeof(custnumb), &cbData);
        SQLGetData(hStmt, 5, SQL_C_CHAR, custname, sizeof(custname), &cbData);

        printf("\n\t%8s%10s\n", custnumb, custname);
    }
    SQLFreeStmt(hStmt,SQL_CLOSE);
}
When an application and/or process has completed operations with the database, it must release resources by disconnecting from the database engine.

Below is the code for `Done()`:

```c
/*
 * Done()
 *
 * This function handles the housekeeping of closing connection and
 * freeing of associated memory
 */
void Done(void)
{
    RETCODE rc;

    printf("DONE\n");

    /* free statement handle */
    if ((rc = SQLFreeHandle(SQL_HANDLE_STMT, hStmt)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLFreeHandle(SQL_HANDLE_STMT)");

    /* disconnect from server */
    printf("\tLogout...
");
    if ((rc = SQLDisconnect(hDbc)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLDisconnect()");

    /* free connection handle */
    if ((rc = SQLFreeHandle(SQL_HANDLE_DBC, hDbc)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLFreeHandle(SQL_HANDLE_DBC)");

    /* free environment handle */
    if ((rc = SQLFreeHandle(SQL_HANDLE_ENV, hEnv)) != SQL_SUCCESS)
        Handle_Error(0, NULL, "SQLFreeHandle(SQL_HANDLE_ENV)");
}```
Additional Resources

We encourage you to explore the additional resources listed here:

- Complete source code for this tutorial can be found in ODBCTutorial1.c in your installation directory, within the sql.odbc\tutorials directory for your platform. Example for the Windows platform:
  
  C:\FairCom\V\win32\sql.odbc\tutorials\ODBCTutorial1.c

  (If you are using c-treeRTG, adjust the path to match your version, e.g., V2.2.0.RTG, and replace sdk with Driver.)

- Additional documentation may be found on the FairCom Web site at: www.faircom.com
2.2 Relationships

..\sdk\sql.odbc.tutorials\ODBCTutorial2.c

If you are using c-treeRTG, adjust the path to match your version, e.g., V2.2.0.RTG, and replace sdk with Driver.

Now we will build some table/file relationships using the c-treeACE SQL ODBC Interface.

This tutorial will advance the concepts introduced in the first tutorial by expanding the number of tables. We will define key columns/fields and create specific indices for each table to form a relational model database.

Like all other examples in the c-tree tutorial series, this tutorial simplifies the creation and use of a database into four simple steps: Initialize(), Define(), Manage(), and You’re Done()!

Tutorial #2: Relational Model and Indexing

Here we add a bit more complexity, introducing multiple tables, with related indices in order to form a simple "relational" database simulating an Order Entry system. Here is an overview of what will be created:

Relational Model Tables

- Initialize() - Connects to the c-treeACE Database Engine.
- Define() - Defines and creates the "custmast", "custordr", "ordritem" and the "itemmast" tables/files with related indices.
- Manage() - Adds some related rows/records to all tables/files. Then queries the database.
- Done() - Disconnects from c-treeACE Database Engine.

Note our simple mainline:

```c
/*
 * main()
 *
 * The main() function implements the concept of "init, define, manage
 * and you're done..."
 */
```
We suggest opening the source code with your own editor.

Continue now to review these four steps.
First we need to open a connection to a database by providing the c-treeACE Database Engine with a user name, password and the database name.

Below is the code for Initialize():

```c
/*
 * Initialize()
 *
 * Perform the minimum requirement of logging onto the c-tree Server
 */
void Initialize(void)
{
   RETCODE  rc;

   printf("INIT\n");

   /* allocate environment handle */
   if ((rc = SQLAllocEnv(&hEnv)) != SQL_SUCCESS)
      Handle_Error(0, NULL, "SQLAllocEnv()");

   /* allocate connection handle */
   if ((rc = SQLAllocConnect(hEnv, &hDbc)) != SQL_SUCCESS)
      Handle_Error(SQL_HANDLE_ENV, hEnv, "SQLAllocConnect()");

   /* connect to server */
   printf("Logon to server...\n");
   if ((rc = SQLConnect(hDbc, MY_DSN, SQL_NTS, "admin", SQL_NTS, "ADMIN", SQL_NTS)) != SQL_SUCCESS)
      Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLConnect()");

   /* allocate statement handle */
   if ((rc = SQLAllocHandle(SQL_HANDLE_STMT, hDbc, &hStmt)) != SQL_SUCCESS)
      Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLAllocHandle(SQL_HANDLE_STMT)");
}
```
Define

The define step is where specific data definitions are established by your application and/or process. This involves defining columns/fields and creating the tables/files with optional indices.

Below is the code for Define():

```c
/* Define()
* Create the tables */
void Define(void)
{
    printf("DEFINE\n");

    Create_CustomerMaster_Table();
    Create_CustomerOrders_Table();
    Create_OrderItems_Table();
    Create_ItemMaster_Table();
}

/*
* Create_CustomerMaster_Table()
* Create the CustomerMaster */
void Create_CustomerMaster_Table(void)
{
    RETCODE rc;

    /* define table CustomerMaster */
    printf("\ttable CustomerMaster\n");

    if ((rc = SQLExecDirect(hStmt,
        "CREATE TABLE custmast ( \
            cm_custnumb CHAR(4), \ 
            cm_custzipc CHAR(9), \ 
            cm_custstat CHAR(2), \ 
            cm_custrtng CHAR(1), \ 
            cm_custname VARCHAR(47), \ 
            cm_custaddr VARCHAR(47), \ 
            cm_custcity VARCHAR(47))",
        SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(CREATE TABLE)");

    if ((rc = SQLExecDirect(hStmt,
        "CREATE UNIQUE INDEX cm_custnumb_idx ON custmast (cm_custnumb)",
        SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(CREATE INDEX)");
```
/*
 * Create_CustomerOrders_Table()
 * Create the table CustomerOrders
 */
void Create_CustomerOrders_Table(void)
{
    RETCODE rc;

    /* define table CustomerOrders */
    printf("\table CustomerOrders\n");

    if ((rc = SQLExecDirect(hStmt,
        "CREATE TABLE custordr ( \
        co_ordrdate DATE, \
        co_promdate DATE, \
        co_ordrnumb CHAR(6), \
        co_custnumb CHAR(4)),
        SQL_NTS) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(CREATE TABLE)");

    if ((rc = SQLExecDirect(hStmt,
        "CREATE UNIQUE INDEX co_ordrnumb_idx ON custordr (co_ordrnumb)",
        SQL_NTS) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(CREATE INDEX)");

    if ((rc = SQLExecDirect(hStmt,
        "CREATE INDEX co_custnumb_idx ON custordr (co_custnumb)",
        SQL_NTS) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(CREATE INDEX)");
}

/*
 * Create_OrderItems_Table()
 * Create the table OrderItems
 */
void Create_OrderItems_Table(void)
{
    RETCODE rc;

    /* define table OrderItems */
    printf("\table OrderItems\n");

    if ((rc = SQLExecDirect(hStmt,
        "CREATE TABLE ordritem ( \
        oi_sequnumb SMALLINT, \
        oi_quantity SMALLINT, \
        oi_ordrnumb CHAR(6), \
        oi_itemnumb CHAR(5)),
        SQL_NTS) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(CREATE TABLE)");

    if ((rc = SQLExecDirect(hStmt,
        "CREATE UNIQUE INDEX oi_ordrnumb_idx ON ordritem (oi_ordrnumb)",
        SQL_NTS) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(CREATE INDEX)");

    if ((rc = SQLExecDirect(hStmt,
        "CREATE INDEX oi_itemnumb_idx ON ordritem (oi_itemnumb)",
        SQL_NTS) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(CREATE INDEX)");
CREATE TABLE itemmast (im_itemwght INTEGER, im_itempric MONEY, im_itemnumb CHAR(5), im_itemdesc VARCHAR(47))
if ((rc = SQLExecDirect(hStmt, "CREATE TABLE itemmast (im_itemwght INTEGER, im_itempric MONEY, im_itemnumb CHAR(5), im_itemdesc VARCHAR(47))", SQL_NTS)) != SQL_SUCCESS) Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(CREATE TABLE)"
}
Manage

The manage step provides data management functionality for your application and/or process.

Below is the code for `Manage()`:

```c
/*
* Manage()
*
* Populates table and perform a simple query
*
*/
void Manage(void)
{
    RETCODE  rc;
    SQLTCHAR custname[47+1];
    SQLREAL  total;
    SDWORD   cbData;

    printf("MANAGE\n");

    /* populate the tables with data */
    Add_CustomerMaster_Records();
    Add_CustomerOrders_Records();
    Add_OrderItems_Records();
    Add_ItemMaster_Records();

    /* perform a query:
      list customer name and total amount per order
    */
    printf("\nQuery Results\n");
    if ((rc = SQLExecDirect(hStmt,  
        "SELECT cm_custname, SUM(im_itempric * oi_quantity) 
        "FROM custmast, custordr, ordritem, itemmast 
        "WHERE co_custnumb = cm_custnumb AND co_ordrnumb = oi_ordrnumb AND oi_itemnumb = 
        im_itemnumb 
        "GROUP BY cm_custnumb, cm_custname",  
        SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(SELECT)" );
```
/* for each order in the CustomerOrders table */
while ((rc = SQLFetch(hStmt)) == SQL_SUCCESS)
{
  if ((rc = SQLGetData(hStmt, 1, SQL_C_CHAR, custname, sizeof(custname), &cbData)) != SQL_SUCCESS)
    Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLGetData()");
  if ((rc = SQLGetData(hStmt, 2, SQL_C_FLOAT, &total, sizeof(total), &cbData)) != SQL_SUCCESS)
    Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLGetData()");
  /* output data to stdout */
  printf("%20s %.2fn", custname, total);
}
SQLFreeStmt(hStmt, SQL_CLOSE);

/*
 * Add_CustomerMaster_Records()
 *
 * This function adds records to table CustomerMaster from an
 * array of strings
 */
void Add_CustomerMaster_Records(void)
{
  RETCODE      rc;
  SQLINTEGER   i;
  SQLTCHAR     sCommand[512];
  SQLTCHAR     *data[] = {
    "('1000','92867','CA','1','Bryan Williams','2999 Regency','Orange')",
    "('1001','61434','CT','1','Michael Jordan','13 Main','Harford')",
    "('1002','73677','GA','1','Joshua Brown','4356 Cambridge','Atlanta')",
    "('1003','10034','MO','1','Keyon Dooling','19771 Park Avenue','Columbia')"
  };
  SQLINTEGER   nRecords = sizeof(data) / sizeof(data[0]);
  Delete_Records("custmast");
  printf("Add records in table CustomerMaster...
"n);
  /* add one record at time to table */
  for (i = 0; i < nRecords; i++)
  {
    strcpy (sCommand, "INSERT INTO custmast VALUES ");
    strcat (sCommand, data[i]);
    if ((rc = SQLExecDirect(hStmt, sCommand ,SQL_NTS)) != SQL_SUCCESS)
      Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(INSERT)"");
  }
}

/*
 * Add_CustomerOrders_Records()
 *
 * This function adds records to table CustomerOrders from an
 * array of strings
 */
void Add_CustomerOrders_Records(void)
{
  RETCODE   rc;
  SQLINTEGER   i;
SQLTCHAR sCommand[512];
SQLTCHAR *data[] = {
    "('09/01/2002','09/05/2002','1','1001')",
    "('09/02/2002','09/06/2002','2','1002')"
};
SQLINTEGER nRecords = sizeof(data) / sizeof(data[0]);

Delete_Records("custordr");
printf("\tAdd records in table CustomerOrders...\n");
/* add one record at time to table */
for (i = 0; i < nRecords; i++) {
    strcpy (sCommand, "INSERT INTO custordr VALUES ");
    strcat (sCommand, data[i]);
    if ((rc = SQLExecDirect(hStmt, sCommand ,SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(INSERT)"
    }
}

/*
* Add_OrderItems_Records()
* This function adds records to table OrderItems from an
* array of strings
*/
void Add_OrderItems_Records(void)
{
    RETCODE rc;
    SQLINTEGER i;
    SQLTCHAR sCommand[512];
    SQLTCHAR *data[] = {
        "(1,2,'1','1')",
        "(2,1,'1','2')",
        "(3,1,'1','3')",
        "(1,3,'2','3')"
    };
    SQLINTEGER nRecords = sizeof(data) / sizeof(data[0]);
    Delete_Records("ordritem");
    printf("\tAdd records in table OrderItems...\n");
    /* add one record at time to table */
    for (i = 0; i < nRecords; i++) {
        strcpy (sCommand, "INSERT INTO ordritem VALUES ");
        strcat (sCommand, data[i]);
        if ((rc = SQLExecDirect(hStmt, sCommand ,SQL_NTS)) != SQL_SUCCESS)
            Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(INSERT)"
        }
    }
    }
*/
void Add_ItemMaster_Records(void)
{
    RETCODE rc;
    SQLINTEGER i;
    SQLTCHAR sCommand[512];
    SQLTCHAR *data[] = {
        "(10,19.95,'1','Hammer')",
        "(3, 9.99,'2','Wrench')",
        "(4, 16.59,'3','Saw')",
        "(1, 3.98,'4','Pliers')"
    };
    SQLINTEGER nRecords = sizeof(data) / sizeof(data[0]);
    Delete_Records("itemmast");
    printf("\tAdd records in table ItemMaster...\n");
    /* add one record at time to table */
    for (i = 0; i < nRecords; i++)
    {
        strcpy (sCommand, "INSERT INTO itemmast VALUES ");
        strcat (sCommand, data[i]);
        if ((rc = SQLExecDirect(hStmt, sCommand ,SQL_NTS)) != SQL_SUCCESS)
            Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(INSERT)");
    }
}

/*
 * Delete_Records()
 * This function deletes all the records in a tables
 */
void Delete_Records(SQLTCHAR* table)
{
    RETCODE rc;
    SQLTCHAR sCommand[512];
    printf("\tDelete records...
");
    sprintf (sCommand, "DELETE FROM %s", table);
    if ((rc = SQLExecDirect(hStmt, sCommand ,SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(DELETE)");
When an application and/or process has completed operations with the database, it must release resources by disconnecting from the database engine.

Below is the code for `Done()`:

```c
/*
 * Done()
 *
 * This function handles the housekeeping of closing connection and
 * freeing of associated memory
 */
void Done(void)
{
    RETCODE rc;

    printf("DONE\n");

    /* free statement handle */
    if ((rc = SQLFreeHandle(SQL_HANDLE_STMT, hStmt)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLFreeHandle(SQL_HANDLE_STMT)");

    /* disconnect from server */
    printf("Logout...
");
    if ((rc = SQLDisconnect(hDbc)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLDisconnect()");

    /* free connection handle */
    if ((rc = SQLFreeHandle(SQL_HANDLE_DBC, hDbc)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLFreeHandle(SQL_HANDLE_DBC)");

    /* free environment handle */
    if ((rc = SQLFreeHandle(SQL_HANDLE_ENV, hEnv)) != SQL_SUCCESS)
        Handle_Error(0, NULL, "SQLFreeHandle(SQL_HANDLE_ENV)");
}
```
Additional Resources

We encourage you to explore the additional resources listed here:

- Complete source code for this tutorial can be found in ODBCTutorial2.c in your installation directory, within the sql.odbc.tutorials directory for your platform. Example for the Windows platform:

  `C:\FairCom\V\win32\sdk\sql.odbc\tutorials\ODBCTutorial2.c`

  (If you are using c-treeRTG, adjust the path to match your version, e.g., V2.2.0.RTG, and replace `sdk` with `Driver`.)

- Additional documentation may be found on the FairCom Web site at: www.faircom.com
2.3 Record/Row Locking

..\sdk\sql.odbc\tutorials\ODBC\Tutorial3.c

If you are using c-treeRTG, adjust the path to match your version, e.g., V2.2.0.RTG, and replace sdk with Driver.

Now we will explore row/record locks using the c-treeACE SQL ODBC Interface.

The functionality for this tutorial focuses on inserting/adding rows/records, then updating a single row/record in the customer master table under locking control. The application will pause after a LOCK is placed on a row/record. Another instance of this application should then be launched, which will block, waiting on the lock held by the first instance. Pressing the <Enter> key will enable the first instance to proceed. This will result in removing the lock thereby allowing the second instance to continue execution. Launching two processes provides a visual demonstration of the effects of locking and a basis for experimentation on your own.

Like all other examples in the c-tree tutorial series, this tutorial simplifies the creation and use of a database into four simple steps: Initialize(), Define(), Manage(), and you’re Done()!

Tutorial #3: Locking

Here we demonstrate the enforcement of data integrity by introducing record/row "locking".

- Initialize() - Connects to the c-treeACE Database Engine.
- Define() - Defines and creates a "customer master" (custmast) table/file.
- Manage() - Adds a few rows/records; Reads the rows/records back from the database; displays the column/field content. Then demonstrates an update operation under locking control, and a scenario that shows a locking conflict.
- Done() - Disconnects from c-treeACE Database Engine.

Note our simple mainline:

```c
int main(int argc, char* argv[])
{
    Initialize();
    Define();
    Manage();
    Done();

    printf("\nPress <ENTER> key to exit . . .\n");
    getchar();
    return(0);
}
```

We suggest opening the source code with your own editor.

Continue now to review these four steps.
First we need to open a connection to a database by providing the c-treeACE Database Engine with a user name, password and the database name.

Below is the code for Initialize():

```c
/*
 * Initialize()
 * 
 * Perform the minimum requirement of logging onto the c-tree Server
 */
void Initialize(void)
{
    RETCODE rc;

    printf("INIT\n");

    /* allocate environment handle */
    if ((rc = SQLAllocEnv(&hEnv)) != SQL_SUCCESS)
        Handle_Error(0, NULL, "SQLAllocEnv()");

    /* allocate connection handle */
    if ((rc = SQLAllocConnect(hEnv, &hDbc)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_ENV, hEnv, "SQLAllocConnect()");

    /* connect to server */
    printf("Logon to server...\n");
    if ((rc = SQLConnect(hDbc, MY_DSN, SQL_NTS, "admin", SQL_NTS, "ADMIN", SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLConnect()");

    /* disable commit after each single SQL statement */
    if ((rc = SQLSetConnectAttr(hDbc, SQL_ATTR_AUTOCOMMIT, SQL_AUTOCOMMIT_OFF, 0)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLSetConnectAttr(SQL_AUTOCOMMIT_OFF)";

    /* allocate statement handle */
    if ((rc = SQLAllocHandle(SQL_HANDLE_STMT, hDbc, &hStmt)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLAllocHandle(SQL_HANDLE_STMT)";

    }
```
Define

The define step is where specific data definitions are established by your application and/or process. This involves defining columns/fields and creating the tables/files with optional indices.

Below is the code for `Define()`:}

```c
/*
 * Define()
 *
 * Create the table for containing a list of existing customers
 */
void Define(void)
{
    RETCODE rc;

    printf("DEFINE\n");

    /* create table */
    printf("\tCreate table...\n");
    if ((rc = SQLExecDirect(hStmt,
        "CREATE TABLE custmast ( \\
        cm_custnumb CHAR(4), \\
        cm_custzipc CHAR(9), \\
        cm_custstat CHAR(2), \\
        cm_custrtng CHAR(1), \\
        cm_custname VARCHAR(47), \\
        cm_custaddr VARCHAR(47), \\
        cm_custcity VARCHAR(47))",
        SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(CREATE TABLE)");

    if ((rc = SQLExecDirect(hStmt,
        "CREATE UNIQUE INDEX cm_custnumb_idx ON custmast (cm_custnumb)",
        SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(CREATE INDEX)");

    if ((rc = SQLExecDirect(hStmt, "COMMIT WORK", SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(COMMIT WORK)"};
```
Manage

The manage step provides data management functionality for your application and/or process.

Below is the code for `Manage()`:

```c
/*
 * Manage()
 *
 * This function performs simple record functions of add, delete and gets
 */
void Manage(void)
{
    printf("MANAGE\n");
    /* delete any existing records */
    Delete_Records();
    /* populate the table with data */
    Add_CustomerMaster_Records();
    /* display contents of table */
    Display_Records();
    /* update a record under locking control */
    Update_CustomerMaster_Record();
    /* display again after update and effects of lock */
    Display_Records();
}

/*
 * Delete_Records()
 *
 * This function deletes all the records in the table
 */
void Delete_Records(void)
{
    RETCODE rc;
    printf("\tDelete records...\n");
    if ((rc = SQLExecDirect(hStmt, "DELETE FROM custmast", SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(DELETE)");
    if ((rc = SQLExecDirect(hStmt, "COMMIT WORK", SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(COMMIT WORK)");
}

/*
 * Add_CustomerMaster_Records()
 */
```
* This function adds records to a table from an array of strings
*
void Add_CustomerMaster_Records(void)
{
    RETCODE rc;
    SQLINTEGER i;
    SQLTCHAR sCommand[512];
    SQLTCHAR *data[] = {
        "('1000','92867','CA','1','Bryan Williams','29
         99 Regency','Orange')",
        "('1001','61434','CT','1','Michael Jordan','13 Main','Harford')",
        "('1002','73677','GA','1','Joshua Brown','4356 Cambridge','Atlanta')",
        "('1003','10034','MO','1','Keyon Dooling','19771 Park Avenue','Columbia')"
    };
    SQLINTEGER nRecords = sizeof(data) / sizeof(data[0]);

    printf("Add records...
");
    /* add one record at time to table */
    for (i = 0; i < nRecords; i++)
    {
        strcpy (sCommand, "INSERT INTO custmast VALUES ");
        strcat (sCommand, data[i]);
        if ((rc = SQLExecDirect(hStmt, sCommand ,SQL_NTS)) != SQL_SUCCESS)
            Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(INSERT)");
    }

    if ((rc = SQLExecDirect(hStmt, "COMMIT WORK", SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(COMMIT WORK)");
}

/*
* Display_Records()
*
* This function displays the contents of a table.
*/
void Display_Records(void)
{
    RETCODE rc;
    SQLTCHAR custnumb[4+1];
    SQLTCHAR custname[47+1];
    SDWORD cbData;

    printf("Display records...
");
    rc = SQLExecDirect(hStmt, "SELECT * FROM custmast",SQL_NTS);
    if (rc != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(SELECT)");

    /* fetch and display each individual record */
    while ((rc = SQLFetch(hStmt)) == SQL_SUCCESS)
    {
        SQLGetData(hStmt, 1, SQL_C_CHAR, custnumb, sizeof(custnumb), &cbData);
        SQLGetData(hStmt, 5, SQL_C_CHAR, custname, sizeof(custname), &cbData);

        printf("%8s%10s\n", custnumb, custname);
    }

    SQLFreeStmt(hStmt,SQL_CLOSE);
}
void Update_CustomerMaster_Record(void)
{
    RETCODE rc;

    printf("\tUpdate record...\n");

    rc = SQLExecDirect(hStmt, "UPDATE custmast SET cm_custname = 'KEYON DOOLING' WHERE cm_custnumb = '1003'", SQL_NTS);
    if (rc != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(UPDATE)");

    printf("\tPress <ENTER> key to unlock\n");
    getchar();

    if ((rc = SQLExecDirect(hStmt, "COMMIT WORK", SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(COMMIT WORK)");

    SQLFreeStmt(hStmt, SQL_CLOSE);
}
When an application and/or process has completed operations with the database, it must release resources by disconnecting from the database engine.

Below is the code for `Done()`:

```c
/*
 * Done()
 * This function handles the housekeeping of closing connection and
 * freeing of associated memory
 */
void Done(void)
{
    RETCODE rc;

    printf("DONE\n");

    /* free statement handle */
    if ((rc = SQLFreeHandle(SQL_HANDLE_STMT, hStmt)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLFreeHandle(SQL_HANDLE_STMT)");

    /* re-enable autocommit */
    if ((rc = SQLSetConnectAttr(hDbc, SQL_ATTR_AUTOCOMMIT, (void *)SQL_AUTOCOMMIT_ON, 0)) !=
        SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLSetConnectAttr(SQL_AUTOCOMMIT_ON)");

    /* disconnect from server */
    printf("\tLogout...\n");
    if ((rc = SQLDisconnect(hDbc)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLDisconnect()"); 

    /* free connection handle */
    if ((rc = SQLFreeHandle(SQL_HANDLE_DBC, hDbc)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_ENV, hEnv, "SQLFreeHandle(SQL_HANDLE_DBC)");

    /* free environment handle */
    if ((rc = SQLFreeHandle(SQL_HANDLE_ENV, hEnv)) != SQL_SUCCESS)
        Handle_Error(0, NULL, "SQLFreeHandle(SQL_HANDLE_ENV)");
}
```
Additional Resources

We encourage you to explore the additional resources listed here:

- Complete source code for this tutorial can be found in ODBCTutorial3.c in your installation directory, within the `sql.odbc\tutorials` directory for your platform. Example for the Windows platform:
  
  `C:\FairCom\V*\win32\sql.odbc\tutorials\ODBCTutorial3.c`

  (If you are using c-treeRTG, adjust the path to match your version, e.g., `V2.2.0.RTG`, and replace `sdk` with `Driver`.)

- Additional documentation may be found on the FairCom Web site at: www.faircom.com
2.4 Transaction Processing

..\sdk\sql.odbc.tutorials\ODBC\Tutorial4.c

If you are using c-treeRTG, adjust the path to match your version, e.g., V2.2.0.RTG, and replace sdk with Driver.

Now we will discuss transaction processing as it relates to the c-treeACE SQL ODBC Interface.

Transaction processing provides a safe method by which multiple database operations spread across separate tables/files are guaranteed to be atomic. By atomic, we mean that, within a transaction, either all of the operations succeed or none of the operations succeed. This "either all or none" atomicity insures that the integrity of the data in related tables/files is secure.

Like all other examples in the c-tree tutorial series, this tutorial simplifies the creation and use of a database into four simple steps: Initialize(), Define(), Manage(), and You're Done()!

Tutorial #4: Transaction Processing

Here we demonstrate transaction control.

- Initialize() - Connects to the c-treeACE Database Engine.
- Define() - Defines and creates our four tables/files.
- Manage() - Adds rows/records to multiple tables/files under transaction control.
- Done() - Disconnects from c-treeACE Database Engine.

Note our simple mainline:

```c
/*
 * main()
 */

int main(int argc, char* argv[])
{
    Initialize();
    Define();
    Manage();
    Done();

    printf("\nPress <ENTER> key to exit . . .\n");
    getchar();
    return(0);
}
```

We suggest opening the source code with your own editor.

Continue now to review these four steps.
First we need to open a connection to a database by providing the c-treeACE Database Engine with a user name, password and the database name.

Below is the code for Initialize():

```c
/*
 * Initialize()
 * Perform the minimum requirement of logging onto the c-tree Server
 */
void Initialize(void)
{
    RETCODE    rc;

    printf("INIT\n");

    /* allocate environment handle */
    if ((rc = SQLAllocEnv(&hEnv)) != SQL_SUCCESS)
        Handle_Error(0, NULL, "SQLAllocEnv()");

    /* allocate connection handle */
    if ((rc = SQLAllocConnect(hEnv, &hDbc)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_ENV, hEnv, "SQLAllocConnect()");

    /* connect to server */
    printf("Logon to server...
");
    if ((rc = SQLConnect(hDbc, MY_DSN, SQL_NTS, "admin", SQL_NTS, "ADMIN", SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLConnect()");

    /* disable commit after each single SQL statement */
    if ((rc = SQLSetConnectAttr(hDbc, SQL_ATTR_AUTOCOMMIT, SQL_AUTOCOMMIT_OFF, 0)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLSetConnectAttr(SQL_AUTOCOMMIT_OFF)");

    /* allocate statement handle */
    if ((rc = SQLAllocHandle(SQL_HANDLE_STMT, hDbc, &hStmt)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLAllocHandle(SQL_HANDLE_STMT"));
}
Define

The define step is where specific data definitions are established by your application and/or process. This involves defining columns/fields and creating the tables/files with optional indices.

Below is the code for Define():

```c
/*
 * Define()
 * Create the tables
 */
void Define(void)
{
    RETCODE rc;

    printf("DEFINE\n");

    /* delete tables... */
    Delete_Tables();
    /* ...and re-create them with constraints */
    Create_CustomerMaster_Table();
    Create_ItemMaster_Table();
    Create_CustomerOrders_Table();
    Create_OrderItems_Table();

    if ((rc = SQLExecDirect(hStmt, "COMMIT WORK", SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(COMMIT WORK)");
}

/*
 * Create_CustomerMaster_Table()
 * Create the table CustomerMaster
 */
void Create_CustomerMaster_Table(void)
{
    RETCODE rc;

    /* define table CustomerMaster */
    printf("\ttable CustomerMaster\n");

    if ((rc = SQLExecDirect(hStmt, "CREATE TABLE custmast ( \n        cm_custnumb CHAR(4) PRIMARY KEY, \n        cm_custzipc CHAR(9), \n        cm_custstat CHAR(2), \n        cm_custrtng CHAR(1), \n        cm_custname VARCHAR(47), \n        cm_custaddr VARCHAR(47), \n        cm_custcity VARCHAR(47))", SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(CREATE TABLE)");
```

```
/*
*  Create_CustomerOrders_Table()
*  Create the table CustomerOrders
*/
void Create_CustomerOrders_Table(void)
{
    RETCODE   rc;

    /* define table CustomerOrders */
    printf("\table CustomerOrders\n");

    if ((rc = SQLExecDirect(hStmt,
        "CREATE TABLE custordr ( \\
            co_ordrdate DATE, \\
            co_promdate DATE, \\
            co_ordrnumb CHAR(6) PRIMARY KEY, \\
            co_custnumb CHAR(4), \\
            FOREIGN KEY (co_custnumb) REFERENCES custmast",
        SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(CREATE TABLE)");

    /* define table OrderItems */
    printf("\table OrderItems\n");

    if ((rc = SQLExecDirect(hStmt,
        "CREATE TABLE ordritem ( \\
            oi_sequnumb SMALLINT, \\
            oi_quantity SMALLINT, \\
            oi_ordrnumb CHAR(6), \\
            oi_itemnumb CHAR(5), \\
            FOREIGN KEY (oi_itemnumb) REFERENCES itemmast, \\
            FOREIGN KEY (oi_ordrnumb) REFERENCES custordr",
        SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(CREATE TABLE)");

}
/* define table ItemMaster */
printf("\table ItemMaster\n");

if ((rc = SQLExecDirect(hStmt,
    "CREATE TABLE itemmast {
       im_itemwght INTEGER,
       im_itempric MONEY,
       im_itemnumb CHAR(5) PRIMARY KEY,
       im_itemdesc VARCHAR(47))",
    SQL_NTS)) != SQL_SUCCESS)
    Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(CREATE TABLE)");
}
The manage step provides data management functionality for your application and/or process.

Below is the code for `Manage()`:

```c
/*
 * Manage()
 * Populates table and performs a simple query
 */
void Manage(void)
{
    printf("MANAGE\n");
    /* populate the tables with data */
    Add_CustomerMaster_Records();
    Add_ItemMaster_Records();
    Add_Transactions();
    /* display the orders and their items */
    Display_CustomerOrders();
    Display_OrderItems();
}

/*
 * Delete_Tables()
 * This function removes all existing tables
 */
void Delete_Tables(void)
{
    RETCODE   rc;
    if ((rc = SQLExecDirect(hStmt, "DROP TABLE ordritem", SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(DROP TABLE)");
    if ((rc = SQLExecDirect(hStmt, "DROP TABLE custordr", SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(DROP TABLE)");
    if ((rc = SQLExecDirect(hStmt, "DROP TABLE custmast", SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(DROP TABLE)");
    if ((rc = SQLExecDirect(hStmt, "DROP TABLE itemmast", SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(DROP TABLE)");
}

/*
 * Add_Transactions()
 * Add an Order and associated Items "as a transaction" to their respective tables. A transaction is committed or aborted if the
* customer number on the order is confirmed valid. Likewise each
* item in the order is verified to be a valid item.
*/
typedef struct {
    SQLTCHAR       *ordrdate, *promdate, *ordrnumb, *custnumb;
} ORDER_DATA;
typedef struct {
    SQLTCHAR       *ordrnumb;
    SQLUSMALLINT   sequnumb;
    SQLUSMALLINT   quantity;
    SQLTCHAR       *itemnumb;
} ORDERITEM_DATA;
ORDER_DATA orders[] = {
    {"09/01/2002", "09/05/2002", "1", "1001"},
    {"09/02/2002", "09/06/2002", "2", "9999"}, /* bad customer number */
    {"09/22/2002", "09/26/2002", "3", "1003"}
};
ORDERITEM_DATA items[] = {
    {"1", 1, 2, "1"},
    {"1", 2, 1, "2"},
    {"2", 1, 3, "3"},
    {"2", 2, 3, "4"},
    {"3", 1, 2, "3"},
    {"3", 2, 2, "99"} /* bad item number */
};

void Add_Transactions(void)
{
    RETCODE     rc;
    SQLINTEGER  i, j = 0;
    SQLINTEGER  nOrders = sizeof(orders) / sizeof(ORDER_DATA);
    SQLINTEGER  nItems = sizeof(items) / sizeof(ORDERITEM_DATA);
    SQLTCHAR    sCommand[512];

    printf("\tAdd transaction records... \n");
    for (i = 0; i < nOrders; i++)
    {
        /* add order record */
        sprintf(sCommand, "INSERT INTO custordr VALUES ('%s', '%s', '%s', '%s')",
                orders[i].ordrdate,
                orders[i].promdate,
                orders[i].ordrnumb,
                orders[i].custnumb);
        if ((rc = SQLExecDirect(hStmt, sCommand, SQL_NTS)) != SQL_SUCCESS)
            Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(INSERT)\n");

        /* process order items */
        while (!strcmp(items[j].ordrnumb, orders[i].ordrnumb))
        {
            /* add item record */
            sprintf(sCommand, "INSERT INTO ordritem VALUES (%d, %d, '%s', '%s')",
                    items[j].sequnumb,
                    items[j].quantity,
                    items[j].ordrnumb,
                    items[j].itemnumb;
            if ((rc = SQLExecDirect(hStmt, sCommand, SQL_NTS)) != SQL_SUCCESS)
                Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(INSERT)\n");
        }
    }
}
/* bump to next item */
  j++;

/* exit the while loop on last item */
if (j >= nItems)
  break;

/* commit the transaction */
if ((rc = SQLExecDirect(hStmt, "COMMIT WORK", SQL_NTS)) != SQL_SUCCESS)
  Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(COMMIT WORK)");


/* Add_CustomerMaster_Records() */
* This function adds records to table CustomerMaster from an
* array of strings
*/
void Add_CustomerMaster_Records(void)
{
  RETCODE rc;
  SQLINTEGER i;
  SQLTCHAR sCommand[512];
  SQLTCHAR *data[] = {
    "('1000','92867','CA','1','Bryan Williams','2999 Regency','Orange')",
    "('1001','61434','CT','1','Michael Jordan','13 Main','Harford')",
    "('1002','73677','GA','1','Joshua Brown','4356 Cambridge','Atlanta')",
    "('1003','10034','MO','1','Keyon Dooling','19771 Park Avenue','Columbia')"
  };
  SQLINTEGER nRecords = sizeof(data) / sizeof(data[0]);
  printf("\tAdd records in table CustomerMaster...\n");

  /* add one record at time to table */
  for (i = 0; i < nRecords; i++)
    {
    strcpy(sCommand, "INSERT INTO custmast VALUES ");
    strcat(sCommand, data[i]);
    if ((rc = SQLExecDirect(hStmt, sCommand, SQL_NTS)) != SQL_SUCCESS)
      Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(INSERT)");
  }

  if ((rc = SQLExecDirect(hStmt, "COMMIT WORK", SQL_NTS)) != SQL_SUCCESS)
    Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(COMMIT WORK)");

}

/* Add_ItemMaster_Records() */
* This function adds records to table ItemMaster from an
* array of strings
*/
void Add_ItemMaster_Records(void)
{
  RETCODE rc;
  SQLINTEGER i;

  /* add one record at time to table */
  for (i = 0; i < nRecords; i++)
    {
    strcpy(sCommand, "INSERT INTO itemmast VALUES ");
    strcat(sCommand, data[i]);
    if ((rc = SQLExecDirect(hStmt, sCommand, SQL_NTS)) != SQL_SUCCESS)
      Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(INSERT)");
  }

  if ((rc = SQLExecDirect(hStmt, "COMMIT WORK", SQL_NTS)) != SQL_SUCCESS)
    Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(COMMIT WORK)");

}
SQLTCHAR sCommand[512];
SQLTCHAR *data[] = {
    "(10, 19.95,'1','Hammer')",
    "(3, 9.99,'2','Wrench')",
    "(4, 16.59,'3','Saw')",
    "(1, 3.98,'4','Pliers')"
};
SQLINTEGER nRecords = sizeof(data) / sizeof(data[0]);

printf("\tAdd records in table ItemMaster...\n");

/* add one record at time to table */
for (i = 0; i < nRecords; i++)
{
    strcpy (sCommand, "INSERT INTO itemmast VALUES ");
    strcat (sCommand, data[i]);
    if ((rc = SQLExecDirect(hStmt, sCommand, SQL_NTS)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(INSERT)\n");
}

if ((rc = SQLExecDirect(hStmt, "COMMIT WORK", SQL_NTS)) != SQL_SUCCESS)
    Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(COMMIT WORK)\n");


/*
 * Display_CustomerOrders()
 * This function displays the contents of CustomerOrders table
 */
void Display_CustomerOrders(void)
{
    RETCODE rc;
    SQLTCHAR ordrnumb[6+1], custnumb[4+1];
    SDWORD cbData;

    printf("\n\tCustomerOrders Table...
");

    rc = SQLExecDirect(hStmt, "SELECT * FROM custordr", SQL_NTS);
    if (rc != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(SELECT)\n");

    /* fetch and display each individual record */
    while ((rc = SQLFetch(hStmt)) == SQL_SUCCESS)
    {
        SQLGetData(hStmt, 3, SQL_C_CHAR, ordrnumb, sizeof(ordrnumb), &cbData);
        SQLGetData(hStmt, 4, SQL_C_CHAR, custnumb, sizeof(custnumb), &cbData);
        printf("\t %s %s\n", ordrnumb, custnumb);
    }

    SQLFreeStmt(hStmt, SQL_CLOSE);
}

/*
 * Display_OrderItems()
 * This function displays the contents of OrderItems table
 */
void Display_OrderItems(void)
{
RETCODE rc;
SQLTCHAR ordrnumb[6+1], itemnumb[5+1];
SDWORD cbData;

printf("\n\tOrderItems Table...\n");

rc = SQLExecDirect(hStmt, "SELECT * FROM ordritem", SQL_NTS);
if (rc != SQL_SUCCESS)
    Handle_Error(SQL_HANDLE_STMT, hStmt, "SQLExecDirect(SELECT)" );

/* fetch and display each individual record */
while ((rc = SQLFetch(hStmt)) == SQL_SUCCESS)
{
    SQLGetData(hStmt, 3, SQL_C_CHAR, ordrnumb, sizeof(ordrnumb), &cbData);
    SQLGetData(hStmt, 4, SQL_C_CHAR, itemnumb, sizeof(itemnumb), &cbData);
    printf("\t %s   %s
", ordrnumb, itemnumb);
}

SQLFreeStmt(hStmt, SQL_CLOSE);
When an application and/or process has completed operations with the database, it must release resources by disconnecting from the database engine.

Below is the code for `Done()`:

```c
/*
 * Done()
 * This function handles the housekeeping of closing connection and
 * freeing of associated memory
 */
void Done(void)
{
    RETCODE rc;

    printf("DONE\n");

    /* re-enable autocommit */
    if ((rc = SQLSetConnectAttr(hDbc, SQL_ATTR_AUTOCOMMIT, (void *)SQL_AUTOCOMMIT_ON, 0)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLSetConnectAttr(SQL_AUTOCOMMIT_ON)");

    Delete_Tables();

    /* free statement handles */
    if ((rc = SQLFreeHandle(SQL_HANDLE_STMT, hStmt)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLFreeHandle(SQL_HANDLE_STMT)");

    /* disconnect from server */
    printf("\tLogout...\n");
    if ((rc = SQLDisconnect(hDbc)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLDisconnect()");

    /* free connection handle */
    if ((rc = SQLFreeHandle(SQL_HANDLE_DBC, hDbc)) != SQL_SUCCESS)
        Handle_Error(SQL_HANDLE_DBC, hDbc, "SQLFreeHandle(SQL_HANDLE_DBC)");

    /* free environment handle */
    if ((rc = SQLFreeHandle(SQL_HANDLE_ENV, hEnv)) != SQL_SUCCESS)
        Handle_Error(0, NULL, "SQLFreeHandle(SQL_HANDLE_ENV)");
}
```
Additional Resources

We encourage you to explore the additional resources listed here:

- Complete source code for this tutorial can be found in ODBCTutorial4.c in your installation directory, within the `sql.odbc\tutorials` directory for your platform. Example for the Windows platform:
  
  `C:\FairCom\V*\win32\sql.odbc\tutorials\ODBCTutorial4.c`

  (If you are using c-treeRTG, adjust the path to match your version, e.g., `V2.2.0.RTG`, and replace `sdk` with `Driver`.)

- Additional documentation may be found on the FairCom Web site at: www.faircom.com
3. Configuring c-treeACE SQL Data Sources

3.1 Introduction

This chapter describes:

- How to add, modify and delete c-treeACE SQL ODBC data sources on Windows client systems using the ODBC Data Source Administrator utility
- The format of the driver and data source information maintained by the ODBC Data Source Administrator

3.2 Configuring Data Sources with the ODBC Data Source Administrator

The ODBC Data Source Administrator is a Microsoft utility to configure ODBC data sources and drivers.

ODBC uses the term "data source" to refer all the information an application needs to connect to a particular database. This information includes the driver name and location, network address, network software, and database name. Depending on the version of Windows, the ODBC Data Source Administrator stores data source information in text files or in the system registry.

The ODBC Data Source Administrator lets you enter the connection details for accessing a c-treeACE database and associate it with a data source name that users refer to when they need to access the data through an ODBC application.
Invoking the ODBC Data Source Administrator Utility

The c-treeACE ODBC driver installation procedure installs the ODBC Driver Manager software and the associated ODBC Data Source Administrator utility if it was not already installed. To configure the ODBC data source, invoke the ODBC Data Source Administrator Utility:

- On Windows 8 and newer, press the Windows key on your keyboard, type in "ODBC Data Sources," and press Enter. This will load the ODBC Data Sources Control Panel.
- On other versions of Windows, select Start > Settings > Control Panel from the Windows menu and double-click the ODBC Data Sources option in the Control Panel.

c-treeACE ODBC Data Sources

Once you invoke the ODBC Data Source Administrator:

1. In the dialog box for the type of data source you choose, choose the Add button. The Add Data Source dialog box appears.
2. Select c-treeACE SQL from the list of installed drivers and choose Finish. The c-treeACE ODBC Setup dialog box appears.
3. Fill in the dialog box fields as shown in the following figure and choose OK. The ODBC Data Source Administrator writes the values you supply to ODBC.INI or to the DSN file you indicated.

**c-treeACE ODBC Setup**

Enter data source name and desired options, then choose OK.

- **Data Source Name**: c-treeACE ODBC Driver
- **Description**: c-treeACE ODBC Driver
- **Host**: localhost
- **Database**: ctreeSQL
- **User ID**: admin
- **Password**: ********
- **Service**: 6597
- **Default Fetch Size**: 
- **Default Query Timeout**: 
- **Preserve Cursor**: OFF
- **Scrollable Cursor**: OFF
- **Client Character Set**: "UTF8" (CP_1P_UTF8)
- **SSL**: c:\certs\client.pem
- **Options**: 

**Data Source Name** - A local name for the c-treeACE SQL data source for use in connect calls and by the ODBC Administrator.

**Description** - Optional descriptive text.

**Host** - Specify the machine name on which the c-treeACE SQL Server is running.

**Database** - The name of the database where the c-treeACE SQL data source resides.

**User ID / Password** - User name and password for connecting to the database. The driver uses those values if the application does not supply them in the call. You can leave these fields blank if you want the driver to use the defaults on the server. If no defaults are defined and you leave these fields blank, the user will be prompted when the application connects.

**Service** - The name of the Service c-treeACE SQL listens to. If empty, sqlnw is used.

**Default Fetch Size** - This value is the size (in bytes) used by the driver to fetch multiple rows from the server. It reduces network requests resulting in performance gains. If not set, the internal buffer size is 5000 bytes.

- In your connection string, set the attribute "FETCH_SIZE=[number of bytes]"
- In your ODBC.INI file, set the attribute "Default Fetch Size=[number of bytes]"
Connection string settings take precedence over DSN and **ODBC.INI** settings.

**Default Query Timeout** - It is possible to set the Default Query Timeout (in seconds) in the DSN and in the connection string.

- In the connection string, the attribute is: `QUERY_TIMEOUT=[number of seconds]`
- In the **ODBC.INI** file, the attribute is: `Default Query Timeout=[number of seconds]`

Settings in the connection string take precedence over the setting in the DSN or **ODBC.INI**.

**Preserve Cursor** - This behavior is configurable from within the ODBC administrator.

**Scrollable Cursor** - This allows you to enable support for ODBC applications that require a scrollable cursor.

**Client Character Set** - Use the drop-down list to select the character set used by the client.

**SSL** - Enter optional parameters to configure the SSL. The following entries can be used to configure the SSL connection:

1) **empty** - Do not use SSL.
2) **BASIC** - Use SSL without certificate checking.
3) `<certificate file name>` - Use SSL with certificate checking using the certificate file specified. If no path is entered, the file must be in the current working directory.

**Options** - Enter any optional parameters to be included in the connect string.

4. The Data Source Dialog box reappears, and now includes the newly-added data source.

### Configuring SSL in Your Connection String

In the ODBC connection string, it is possible to add "SSL=xyz" where "xyz" is one of the options 2 or 3 from the SSL parameters listed above in the c-treeACE ODBC Setup dialog box.

You can add one of the following to your existing ODBC connection strings to enable TLS/SSL:

- "SSL=BASIC" - Encryption with default server certificate.
- "SSL=ctsrvr.pem" - Use Peer authentication with explicitly named cert located in the local directory.
- "SSL=C:\certs\ctsrvr.pem" - Use Peer authentication with an explicitly named cert with a full path.

### Modifying and Deleting c-treeACE SQL Data Sources

You can modify or delete a c-treeACE SQL data source after you add it. Invoke the ODBC Data Source Administrator and select the data source you want to modify or delete. Then choose the **Configure** or **Remove** button:

- Choosing the **Configure** button displays the c-treeACE ODBC Setup dialog box (see the previous figure) with the current values for that data source. Change the values of any fields, then choose **OK**.

  When you modify a data source, the ODBC Data Source Administrator modifies the entry for the data source name you specify. For example, you could modify a c-treeACE SQL data source to change the user name and password that connections use.

- Choosing the **Remove** button displays a confirmation box. Choose **OK** if you want to delete the data source.
When you delete a data source, the ODBC Data Source Administrator deletes the entry for that data source. (Deleting a data source has no effect on any database, only on ODBC's information for accessing the database.)

### 3.3 ODBC Data Source and Driver Information FORMAT

#### Windows Registry Entries

ODBC driver information is stored in subkeys of the Windows registry.

User data sources are stored under the HKEY_CURRENT_USER key and are available only to the current user. System data sources are stored under the HKEY_LOCAL_MACHINE key and can be used by more than one user.

For user data sources, the complete registry key is HKEY_CURRENT_USER\Software\ODBC\ODBC.INI.

The `ODBCINST.INI` subkey is a subkey of HKEY_LOCAL_MACHINE, which describes the number and types of ODBC drivers installed on the system. The ODBC Data Source Administrator reads the subkey when it invokes the driver’s setup DLL to add or modify a data source for the driver.

### 3.4 Query Timeout Options

c-treeACE SQL supports a timeout option for an executing query. This feature can ensure that an unintended query statement does not consume excessive processing time.

With c-treeACE ODBC you can set the query timeout value for the statement with the `SQLSetStmtAttr()` c-treeACE ODBC API function and the `SQL_ATTR_QUERY_TIMEOUT` parameter set to the number of seconds to wait for the query to execute before returning to the application. A value of 0 indicates no timeout value, which is also the default. The following example code will set a query timeout value of five seconds for the referenced statement handle.

**ODBC Example**

```c
/* Set the Query timeout to 5 seconds */
SQLSetStmtAttr(hstmt, (void*)SQL_ATTR_QUERY_TIMEOUT, 5, 0);
```

Using ODBC through ADO.NET, you can specify the `OdbcConnection.CommandTimeout` property to set a query timeout value on an ODBC statement as demonstrated with the following syntax.

**ODBC via ADO.NET Example**

```c
OdbcConnection myConnection = new OdbcConnection();
myConnection.ConnectionString = "DSN=c-treeSQL ODBC Database";
myConnection.Open();

OdbcCommand oc = new OdbcCommand("SELECT TOP 50000 FROM my_big_table WHERE this < that AND this_string = 'that_string' ORDER BY foo", myConnection);

// Set a query timeout of 5 seconds.
oc.CommandTimeout = 5;
```
try
{
    oc.ExecuteReader();
}
catch (Exception ex)
{
    // Log some error
}

3.5 ODBC Driver Socket SEND/RECV Timeout

A send/recv timeout option is available such that a c-treeACE ODBC client can request a timeout for a connection socket. If the client experiences a lengthy wait for the server to reply, the client can continue to work after closing the connection.

A c-treeACE ODBC driver can set the timeout with a call to the SQLSetConnectAttr() ODBC API function and the SQL_ATTR_CONNECTION_TIMEOUT parameter with the time value in seconds.

Example
/* Set socket timeout to 5 seconds. */
SQLSetConnectAttr(hdbc, (void*)SQL_ATTR_CONNECTION_TIMEOUT, 5, 0);

Note: SQLSetConnectAttr(SQL_ATTR_CONNECTION_TIMEOUT) sets the timeout value for the entire ODBC driver, not just for the current connection. A default value of 0 indicates no timeout.

3.6 Driver Login Timeout

The c-treeACE ODBC Driver for Windows supports a configurable timeout on driver connection login. An application can set the login timeout by calling the SQLSetConnectAttr() ODBC API function with the SQL_ATTR_LOGIN_TIMEOUT attribute and a timeout value.

/* Set the Login timeout to 5 seconds */
SQLSetConnectAttr(hdbc, (void*)SQL_ATTR_LOGIN_TIMEOUT, 5, 0);

This timeout sets the maximum time for which the ODBC driver waits for the connection attempt to c-treeACE SQL to complete. If the connection attempt does not complete in the specified time period, the ODBC driver fails the connection attempt with the following error:
20212, "Error in Network Daemon".

The default login timeout value is 15 seconds.
3.7 Using Reserved Keywords with Microsoft Excel

A feature was added to allow a query from Microsoft Excel and ODBC on tables containing fields identified with c-treeACE SQL reserved keywords. The c-treeACE ODBC Driver now considers a qualified identifier (i.e., a field name preceded by the table name in the form: tablename.fieldname) as a field even if it is a reserved keyword. This is done by automatically wrapping the qualified fieldname in double quotes, which has the effect of making it case-sensitive. Because this behavior only applies to qualified identifiers it can be avoided by specifying only the fieldname without the tablename.

To use this particular feature with Microsoft Excel and the c-treeACE ODBC Driver, a data source should specify the string "DHQQ1" in the Options field of the Data Source Configuration window. This option enables the extended processing for only this connection.
4. ODBC Conformance Notes

4.1 Introduction

This chapter details the ODBC functionality that c-treeACE SQL supports through the ODBC Driver.

4.2 General ODBC and SQL Support Levels

ODBC specifies general conformance levels in two areas:

- ODBC application programming interface (API). c-treeACE SQL supports all Core and Level 1 ODBC API functions, and most Level 2 functions.
- ODBC SQL syntax. c-treeACE SQL supports Extended SQL syntax.

The rest of this chapter provides more detail on that support. Specifically, it details the information the c-treeACE ODBC driver returns when applications call the following functions:

- **SQLGetInfo()** returns various details about the driver and its data source
- Environment, connection, and statement attribute functions return or specify attributes of driver behavior
- **SQLGetFunctions()** returns ODBC functions the driver supports
- **SQLGetTypeInfo()** returns data types the driver supports

4.3 Responses to SQLGetInfo

Applications call the **SQLGetInfo()** function to retrieve details about support for a specific driver and data source provided for different ODBC functionality.

Applications supply the **InfoType** argument to **SQLGetInfo()** to specify what information type they seek. **SQLGetInfo()** returns the information to the **InfoValuePtr** output argument. The following table lists each **InfoType** argument the ODBC Driver recognizes along with the associated value returned to **SQLGetInfo()**. Shaded rows indicate information types renamed or deprecated for ODBC 3.0

For more detail about the **SQLGetInfo()** function see the Microsoft ODBC Programmer’s Reference.
### Information the ODBC Driver Returns to `SQLGetInfo()`

<table>
<thead>
<tr>
<th>Description</th>
<th>InfoType Argument</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guaranteed execute privileges on all procedures returned by SQLProcedures</td>
<td>SQL_ACCESSIBLE_PROCEDURES</td>
<td>Y (guaranteed)</td>
</tr>
<tr>
<td>Guaranteed read access to all table names returned by SQLTables</td>
<td>SQL_ACCESSIBLE_TABLES</td>
<td>N (not guaranteed)</td>
</tr>
<tr>
<td>Maximum number of active connections</td>
<td>SQL_ACTIVE_CONNECTIONS (Renamed for ODBC 3.0.)</td>
<td>See SQL_MAX_DRIVER_CONNECTIONS</td>
</tr>
<tr>
<td>Maximum number of active environments</td>
<td>SQL_ACTIVE_ENVIRONMENTS (ODBC 3.0)</td>
<td>0 (no limit)</td>
</tr>
<tr>
<td>Maximum number of active SQL statements</td>
<td>SQL_ACTIVE_STATEMENTS (Renamed for ODBC 3.0.)</td>
<td>See SQL_MAX_CONCURRENT_ACTIVITIES</td>
</tr>
<tr>
<td>Aggregate function support</td>
<td>SQL_AGGREGATE_FUNCTIONS (ODBC 3.0)</td>
<td>SQL_AF_ALL</td>
</tr>
<tr>
<td>Support for ALTER DOMAIN statement</td>
<td>SQL_ALTER_DOMAIN (ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Support for ALTER TABLE clauses</td>
<td>SQL_ALTER_TABLE</td>
<td>SQL_AT_ADD_COLUMN (supports adding columns)</td>
</tr>
<tr>
<td>Level of asynchronous mode support</td>
<td>SQL_ASYNC_MODE (ODBC 3.0)</td>
<td>SQL_AM_NONE (not supported)</td>
</tr>
<tr>
<td>Behavior with respect to the availability of row counts in batches</td>
<td>SQL_BATCHgetRow_COUNT (ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Support for batches</td>
<td>SQL_BATCH_SUPPORT (ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Support for bookmarks</td>
<td>SQL_BOOKMARK_PERSISTENCE</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Position of qualifier in a qualified table name</td>
<td>SQL_CATALOG_LOCATION (Renamed for ODBC 3.0. Was SQL_QUALIFIER_LOCATION)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Support for catalog names</td>
<td>SQL_CATALOG_NAME (ODBC 3.0)</td>
<td>N (does not support)</td>
</tr>
<tr>
<td>Character used to separate table, column qualifiers</td>
<td>SQL_CATALOG_NAME_SEPARATOR (Renamed for ODBC 3.0. Was SQL_QUALIFIER_NAME_SEPARATOR)</td>
<td>* *</td>
</tr>
<tr>
<td>Description</td>
<td>InfoType Argument</td>
<td>Returns</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>--------------------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>Name for a catalog</td>
<td>SQL_CATALOG_TERM (Renames for ODBC 3.0. Was SQL_QUALIFIER_TERM)</td>
<td>“*” (does not support)</td>
</tr>
<tr>
<td>Statements that support catalog names</td>
<td>SQL_CATALOG_USAGE (Renames for ODBC 3.0. Was SQL_QUALIFIER_USAGE)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Default collation sequence name for the default character set</td>
<td>SQL_COLLATION_SEQ (ODBC 3.0)</td>
<td>“*” (unknown)</td>
</tr>
<tr>
<td>Support for column aliases</td>
<td>SQL_COLUMN_ALIAS</td>
<td>Y (supports)</td>
</tr>
<tr>
<td>Result of concatenation of NULL character column with non-NULL column</td>
<td>SQL_CONCAT_NULL_BEHAVIOR</td>
<td>SQL_CB_NULL (result is null)</td>
</tr>
<tr>
<td>Conversion from BIGINT</td>
<td>SQL_CONVERT_BIGINT</td>
<td>SQL_CVT_BIGINT</td>
</tr>
<tr>
<td>Conversion from BINARY</td>
<td>SQL_CONVERT_BINARY</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Conversion from BIT</td>
<td>SQL_CONVERT_BIT</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Conversion from CHAR</td>
<td>SQL_CONVERT_CHAR</td>
<td>SQL_CVT_BIGINT</td>
</tr>
<tr>
<td>Conversion from DATE</td>
<td>SQL_CONVERT_DATE</td>
<td>SQL_CVT_CHAR</td>
</tr>
<tr>
<td>Description</td>
<td>InfoType Argument</td>
<td>Returns</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Conversion from DECIMAL</td>
<td>SQL_CONVERT_DECIMAL</td>
<td>SQL_CVT_BIGINT</td>
</tr>
<tr>
<td>Conversion from DOUBLE</td>
<td>SQL_CONVERT_DOUBLE</td>
<td>SQL_CVT_BIGINT</td>
</tr>
<tr>
<td>Conversion from FLOAT</td>
<td>SQL_CONVERT_FLOAT</td>
<td>SQL_CVT_BIGINT</td>
</tr>
<tr>
<td>Support for conversion functions</td>
<td>SQL_CONVERT_FUNCTIONS</td>
<td>SQL_FN_CVT_CONVERT (supports)</td>
</tr>
<tr>
<td>Conversion from INTEGER</td>
<td>SQL_CONVERT_INTEGER</td>
<td>SQL_CVT_BIGINT</td>
</tr>
<tr>
<td>Conversion from INTERVAL_DAY_TIME</td>
<td>SQL_CONVERT_INTERVAL_DAY_TIME(ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Conversion from INTERVAL_YEAR_MONTH</td>
<td>SQL_CONVERT_INTERVAL_YEAR_MONTH(ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Description</td>
<td>InfoType Argument</td>
<td>Returns</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Conversion from LONGVARBINARY</td>
<td>SQL_CONVERT_LONGVARBINARY</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Conversion from LONGVARCHAR</td>
<td>SQL_CONVERT_LONGVARCHAR</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Conversion from NUMERIC</td>
<td>SQL_CONVERT_NUMERIC</td>
<td>SQL_CVT_BIGINT</td>
</tr>
<tr>
<td>Conversion from REAL</td>
<td>SQL_CONVERT_REAL</td>
<td>SQL_CVT_BIGINT</td>
</tr>
<tr>
<td>Conversion from SMALLINT</td>
<td>SQL_CONVERT_SMALLINT</td>
<td>SQL_CVT_BIGINT</td>
</tr>
<tr>
<td>Conversion from TIME</td>
<td>SQL_CONVERT_TIME</td>
<td>SQL_CVT_CHAR</td>
</tr>
<tr>
<td>Conversion from TIMESTAMP</td>
<td>SQL_CONVERT_TIMESTAMP</td>
<td>SQL_CVT_CHAR</td>
</tr>
<tr>
<td>Description</td>
<td>InfoType Argument</td>
<td>Returns</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Conversion from TINYINT</td>
<td>SQL_CONVERT_TINYINT</td>
<td>SQL_CVT_BIGINT</td>
</tr>
<tr>
<td>Conversion from VARBINARY</td>
<td>SQL_CONVERT_VARBINARY</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Conversion from VARCHAR</td>
<td>SQL_CONVERT_VARCHAR</td>
<td>SQL_CVT_BIGINT</td>
</tr>
<tr>
<td>Conversion from WCHAR</td>
<td>SQL_CONVERT_WCHAR</td>
<td>SQL_CVT_BIGINT</td>
</tr>
<tr>
<td>Conversion from WLONGVARCHAR</td>
<td>SQL_CONVERT_WLONGVARCHAR</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Description</td>
<td>InfoType Argument</td>
<td>Returns</td>
</tr>
<tr>
<td>---------------------------------------------------------------------------</td>
<td>------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Conversion from WVARCHAR</td>
<td>SQL_CONVERT_WVARCHAR</td>
<td>SQL_CVT_BIGINT</td>
</tr>
<tr>
<td>Support for table correlation names</td>
<td>SQL_CORRELATION_NAME</td>
<td>SQL_CN_ANY</td>
</tr>
<tr>
<td>Support for CREATE ASSERTION statement</td>
<td>SQL_CREATE_ASSERTION(ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Support for CREATE CHARACTER SET statement</td>
<td>SQL_CREATE_CHARACTER_SET(ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Support for CREATE COLLATION statement</td>
<td>SQL_CREATE_COLLATION(ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Support for CREATE DOMAIN statement</td>
<td>SQL_CREATE_DOMAIN(ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Support for CREATE SCHEMA statement</td>
<td>SQL_CREATE_SCHEMA(ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Support for CREATE TABLE statement</td>
<td>SQL_CREATE_TABLE(ODBC 3.0)</td>
<td>SQL_CT_CREATE_TABLE</td>
</tr>
<tr>
<td>(supports CREATE TABLE, deleted rows are deleted on commit, supports column constraints and table constraints)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support for CREATE TRANSLATION statement</td>
<td>SQL_CREATE_TRANSLATION (ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Support for CREATE VIEW statement</td>
<td>SQL_CREATE_VIEW(ODBC 3.0)</td>
<td>SQL_CV_CREATE_VIEW</td>
</tr>
<tr>
<td>(supports CREATE VIEW, CHECK OPTION clause)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effect of COMMIT operation on cursors and prepared statements</td>
<td>SQL_CURSOR_COMMIT_BEHAVIOR</td>
<td>SQL_CB_CLOSE (closes cursors but statements remain in prepared state)</td>
</tr>
<tr>
<td>Description</td>
<td>InfoType Argument</td>
<td>Returns</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Effect of ROLLBACK operation on cursors and prepared statements</td>
<td>SQL_CURSOR_ROLLBACK_BEHAVIOR</td>
<td>SQL_CB_CLOSE (closes cursors but statements remain in prepared state)</td>
</tr>
<tr>
<td>Support for cursor sensitivity</td>
<td>SQL_CURSOR_SENSITIVITY(ODBC 3.0)</td>
<td>SQL_INSENSITIVE (all cursors on the statement handle show the result set without reflecting any changes made to it by any other cursor within the same transaction)</td>
</tr>
<tr>
<td>Name of the data source as specified to the ODBC Administrator</td>
<td>SQL_DATA_SOURCE_NAME</td>
<td>(string containing the name)</td>
</tr>
<tr>
<td>Access limited to read-only</td>
<td>SQL_DATA_SOURCE_READ_ONLY</td>
<td>N (read-write access)</td>
</tr>
<tr>
<td>Name of the c-treeSQL data source on the server system</td>
<td>SQL_DATABASE_NAME</td>
<td>(string containing the name)</td>
</tr>
<tr>
<td>Support for date-time literals</td>
<td>SQL_DATETIME LITERALS(ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Name of the database product supporting the data source</td>
<td>SQL_DBMS_NAME</td>
<td>c-treeACE SQL</td>
</tr>
<tr>
<td>Version of the database product</td>
<td>SQL_DBMS_VER</td>
<td>14.00.00.0000</td>
</tr>
<tr>
<td>Support for creation and dropping of indexes</td>
<td>SQL_DDL_INDEX(ODBC 3.0)</td>
<td>SQL_DI_CREATE_INDEX</td>
</tr>
<tr>
<td>Default transaction isolation level</td>
<td>SQL_DEFAULT_TXN_ISOLATION</td>
<td>SQL_TXN_SERIALIZABLE</td>
</tr>
<tr>
<td>Support for describing parameters via DESCRIBE INPUT statement</td>
<td>SQL_DESCRIBE_PARAMETER(ODBC 3.0)</td>
<td>Y (supports)</td>
</tr>
<tr>
<td>Name of the dynamic link library file for the ODBC Driver</td>
<td>SQL_DRIVER_NAME</td>
<td>DHODBC.DLL</td>
</tr>
<tr>
<td>Supported ODBC version</td>
<td>SQL_DRIVER_ODBC_VER</td>
<td>03.52</td>
</tr>
<tr>
<td>Current Version of the ODBC Driver</td>
<td>SQL_DRIVER_VER</td>
<td>14.00.00.0000</td>
</tr>
<tr>
<td>Support for DROP ASSERTION statement</td>
<td>SQL_DROP_ASSERTION(ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Support for DROP CHARACTER SET statement</td>
<td>SQL_DROP_CHARACTER_SET(ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Support for DROP COLLATION statement</td>
<td>SQL_DROP_COLLATION(ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Support for DROP DOMAIN statement</td>
<td>SQL_DROP_DOMAIN(ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Description</td>
<td>InfoType Argument</td>
<td>Returns</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>--------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Support for DROP SCHEMA statement</td>
<td>SQL_DROP_SCHEMA(ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Support for DROP TABLE statement</td>
<td>SQL_DROP_TABLE(ODBC 3.0)</td>
<td>SQL_DT_DROP_TABLE (supports)</td>
</tr>
<tr>
<td>Support for DROP TRANSLATION statement</td>
<td>SQL_DROP_TRANSLATION(ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Support for DROP VIEW statement</td>
<td>SQL_DROP_VIEW(ODBC 3.0)</td>
<td>SQL_DV_DROP_VIEW (supports)</td>
</tr>
<tr>
<td>Supported attributes of a dynamic cursor: subset 1</td>
<td>SQL_DYNAMIC_CURSOR_ATTRIBUTES1(ODBC 3.0)</td>
<td>0 (does not support dynamic cursors)</td>
</tr>
<tr>
<td>Supported attributes of a dynamic cursor: subset 2</td>
<td>SQL_DYNAMIC_CURSOR_ATTRIBUTES2(ODBC 3.0)</td>
<td>SQL_CA2_READ_ONLY_CONCURRENY <strong>SQL_CA2_MAX_ROWS_SELECT</strong> <strong>SQL_CA2_CRC_EXACT</strong></td>
</tr>
<tr>
<td>Support for expressions in ORDER BY clause</td>
<td>SQL_EXPRESSIONS_IN_ORDERBY</td>
<td>Y (supports)</td>
</tr>
<tr>
<td>Direction that FETCH operations can support</td>
<td>SQL_FETCH_DIRECTION(Deprecated in ODBC 3.0.)</td>
<td>SQL_FD_FETCH_NEXT (fetch next row only)</td>
</tr>
<tr>
<td>Single-tier driver behavior</td>
<td>SQL_FILE_USAGE</td>
<td>SQL_FILE_NOT_SUPPORTED (not a single-tier driver)</td>
</tr>
<tr>
<td>Supported attributes of a forward-only cursor: subset 1</td>
<td>SQL_FORWARD_ONLY_CURSOR_ATTRIBUTES1(ODBC 3.0)</td>
<td>SQL_CA1_NEXT (supports SQL_FETCH_NEXT argument to SQLFetchScroll for forward-only cursors)</td>
</tr>
<tr>
<td>Supported attributes of a forward-only cursor: subset 2</td>
<td>SQL_FORWARD_ONLY_CURSOR_ATTRIBUTES2(ODBC 3.0)</td>
<td>SQL_CA2_READ_ONLY_CONCURRENY <strong>SQL_CA2_MAX_ROWS_SELECT</strong> <strong>SQL_CA2_CRC_EXACT</strong></td>
</tr>
<tr>
<td>Supported extensions to SQLGetData</td>
<td>SQL_GETDATA_EXTENSION</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Relationship between GROUP BY clause and columns in the select list</td>
<td>SQL_GROUP_BY</td>
<td>SQL_GB_GROUP_BY_CONTAINS_SELECT (GROUP BY clause must contain all non-aggregated columns in select list)</td>
</tr>
<tr>
<td>Case-sensitivity of user-supplied names</td>
<td>SQL_IDENTIFIER_CASE</td>
<td>SQL_IC_UPPER (Identifiers in SQL are not case-sensitive and are stored in uppercase in system catalog), SQL_IC_LOWER (Identifiers in SQL are not case-sensitive and are stored in lowercase in system catalog)</td>
</tr>
<tr>
<td>Character used to enclose delimited identifiers</td>
<td>SQL_IDENTIFIER_QUOTE_CHAR</td>
<td>** (double quotation mark)</td>
</tr>
<tr>
<td>Supported views in INFORMATION_SCHEMA</td>
<td>SQL_INFO_SCHEMA_VIEWS(ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Description</td>
<td>InfoType Argument</td>
<td>Returns</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>--------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Support for INSERT statement</td>
<td>SQL_INSERT_STATEMENT(ODBC 3.0)</td>
<td>SQL_IS_INSERT_LITERALS</td>
</tr>
<tr>
<td>Referential integrity syntax support</td>
<td>SQL_INTEGRITY(Renamed for ODBC 3.0. Was SQL_ODBC_SQL_OPT_IEF)</td>
<td>Y (supports referential integrity syntax)</td>
</tr>
<tr>
<td>Supported attributes of a keyset cursor: subset 1</td>
<td>SQL_KEYSET_CURSOR_ATTRIBUTES1(ODBC 3.0)</td>
<td>0 (does not support keyset cursors)</td>
</tr>
<tr>
<td>Supported attributes of a keyset cursor: subset 2</td>
<td>SQL_KEYSET_CURSOR_ATTRIBUTES2(ODBC 3.0)</td>
<td>0 (does not support keyset cursors)</td>
</tr>
<tr>
<td>Data-source specific keywords</td>
<td>SQL_KEYWORDS</td>
<td>acos, add_month, after, an, array, ascii, asin, atan, atan2, before, begin, bigint, binary, bind, binding, call, ceiling, chartorowid, chr, cleanup, clustered, colgroup, complex, compress, concat, contains, cos, cot, curdate, current_date, current_user, curtime, cvar, database, datapages, dateadd, datediff, dayname, dayofmonth, dayofweek, dayofyear, db_name, dba, declaration, decode, definition, degrees, dhyte, difference, each, exclusive, exit, exp, explicit, extract, field, file, floor, go, goto, grant, greatest, identified, ifnull, indexpages, initcap, inout, instr, interface, isnull, lastday, lcase, leading, least, length, link, list, localtime, localtimestamp, locate, lock, log, log10, long, lpad, ltrim, lvarchar, main, metadata_only, minus, mod, mode, molty, money, monthename, months_between, newrow, next_day, nocompress, now, nowait, nullvalue, number, nvl, object_id, octet_length, odbc_convert, odbcinfo, oldrow, out, overlay, pctfree, pi, placing, position, power, prefix, quarter, radians, rand, range, raw, referencing, rename, repeat, replace, resource, row, rowid, rowtochar, rpad, rtrim, rownum, searched_case, second, sevice, session_user, share, short, sign, simple_case, sin, sodex, sql_bigint, sql_binary, sql_bit, sql_char, sql_date, sql_decimal, sql_double, sql_float, sql_integer, sql_longvarbinary, sql_longvarchar, sql_numeric, sql_real, sql_smallint, sql_time, sql_timestamp, sql_tinyint, sql_tsi_day, sql_tsi_frac_second, sql_tsi_hour, sql_tsi_minute, sql_tsi_month, sql_tsi_quarter, sql_tsi_second, sql_tsi_week,</td>
</tr>
<tr>
<td>Description</td>
<td>InfoType Argument</td>
<td>Returns</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Support for escape clause in LIKE predicates</td>
<td>SQL_LIKE_ESCAPE_CLAUSE</td>
<td>Y (supports)</td>
</tr>
<tr>
<td>Supported lock types</td>
<td>SQL_LOCK_TYPES(Depricated in ODBC 3.0.)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Maximum number of active concurrent statements in asynchronous mode</td>
<td>SQL_MAX_ASYNC_CONCURRENT_STATEMENTS(ODBC 3.0)</td>
<td>1</td>
</tr>
<tr>
<td>Maximum length in hexadecimal characters of binary literals</td>
<td>SQL_MAX_BINARY_LITERAL_LEN</td>
<td>8192</td>
</tr>
<tr>
<td>Maximum length of a table or column qualifier</td>
<td>SQL_MAX_CATALOG_NAME_LEN(Renamed for ODBC 3.0. Was SQL_MAX_QUALIFIER_NAME_LEN)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>Maximum length in characters of character string literals</td>
<td>SQL_MAX_CHAR_LITERAL_LEN</td>
<td>8192</td>
</tr>
<tr>
<td>Maximum length of a column name</td>
<td>SQL_MAX_COLUMN_NAME_LEN</td>
<td>64</td>
</tr>
<tr>
<td>Maximum number of columns allowed in GROUP BY clause</td>
<td>SQL_MAX_COLUMNS_IN_GROUP_BY</td>
<td>0 (no limit)</td>
</tr>
<tr>
<td>Maximum number of columns allowed in an index</td>
<td>SQL_MAX_COLUMNS_IN_INDEX</td>
<td>100</td>
</tr>
<tr>
<td>Maximum number of columns allowed in ORDER BY clause</td>
<td>SQL_MAX_COLUMNS_IN_ORDER_BY</td>
<td>0 (no limit)</td>
</tr>
<tr>
<td>Maximum number of columns allowed in a select list</td>
<td>SQL_MAX_COLUMNS_IN_SELECT</td>
<td>0 (no limit)</td>
</tr>
<tr>
<td>Maximum number of columns allowed in a table</td>
<td>SQL_MAX_COLUMNS_IN_TABLE</td>
<td>2000</td>
</tr>
<tr>
<td>Maximum number of active SQL statements</td>
<td>SQL_MAX_ACTIVE_STATEMENTS(Renamed for ODBC 3.0. Was SQL_ACTIVE_STATEMENTS)</td>
<td>0 (no maximum)</td>
</tr>
<tr>
<td>Maximum length of a cursor name</td>
<td>SQL_MAX_CURSOR_NAME_LEN</td>
<td>64</td>
</tr>
<tr>
<td>Description</td>
<td>InfoType Argument</td>
<td>Returns</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Maximum number of active connections</td>
<td>SQL_MAX_DRIVER_CONNECTIONS(Renamed for ODBC 3.0. Was SQL_ACTIVE_CONNECTIONS)</td>
<td>0</td>
</tr>
<tr>
<td>Maximum length of user-defined names</td>
<td>SQL_MAX_IDENTIFIER_LEN(ODBC 3.0)</td>
<td>64</td>
</tr>
<tr>
<td>Maximum number of bytes allowed in the combined fields of an index</td>
<td>SQL_MAX_INDEX_SIZE</td>
<td>0 (no limit)</td>
</tr>
<tr>
<td>Maximum length of an owner name</td>
<td>SQL_MAX_OWNER_NAME_LENGTH(Renamed for ODBC 3.0.)</td>
<td>See SQL_MAX_SCHEMA_NAME_LENGTH</td>
</tr>
<tr>
<td>Maximum length of a procedure name</td>
<td>SQL_MAX_PROCEDURE_NAME_LENGTH</td>
<td>64</td>
</tr>
<tr>
<td>Maximum length of a table or column qualifier</td>
<td>SQL_MAX_QUALIFIER_NAME_LENGTH(Renamed for ODBC 3.0.)</td>
<td>See SQL_MAX_CATALOG_NAME_LENGTH</td>
</tr>
<tr>
<td>Maximum length in bytes of a table row</td>
<td>SQL_MAX_ROW_SIZE</td>
<td>0 (no limit)</td>
</tr>
<tr>
<td>Whether maximum row size includes LONGVARCHAR and LONGVARBINARY</td>
<td>SQL_MAX_ROW_SIZE_INCLUDES_LONG</td>
<td>N</td>
</tr>
<tr>
<td>Maximum length of an owner name</td>
<td>SQL_MAX_SCHEMA_NAME_LENGTH(Renamed for ODBC 3.0. Was SQL_MAX_OWNER_NAME_LENGTH)</td>
<td>64</td>
</tr>
<tr>
<td>Maximum number of characters in a SQL statement</td>
<td>SQL_MAX_STATEMENT_LENGTH</td>
<td>Limited by available memory (&lt;2Gb)</td>
</tr>
<tr>
<td>Maximum length of a table name</td>
<td>SQL_MAX_TABLE_NAME_LENGTH</td>
<td>64</td>
</tr>
<tr>
<td>Maximum number of tables allowed in FROM clause</td>
<td>SQL_MAX_TABLES_IN_SELECT</td>
<td>250</td>
</tr>
<tr>
<td>Maximum length of a user name</td>
<td>SQL_MAX_USER_NAME_LENGTH</td>
<td>64</td>
</tr>
<tr>
<td>Support for multiple result sets</td>
<td>SQL_MULT_RESULT_SETS</td>
<td>N (does not support)</td>
</tr>
<tr>
<td>Support for active transactions on multiple connections</td>
<td>SQL_MULTIPLE_ACTIVE_TXN</td>
<td>N</td>
</tr>
<tr>
<td>Whether data source requires length of LONGVARCHAR and LONGVARBINARY data</td>
<td>SQL_NEED_LONG_DATA_LENGTH</td>
<td>N</td>
</tr>
<tr>
<td>Support for NOT NULL clause in CREATE TABLE statement</td>
<td>SQL_NON_NULLABLE_COLUMNS</td>
<td>SQL_NNC_NON_NULL (supports)</td>
</tr>
<tr>
<td>Where null values are sorted in a list</td>
<td>SQL_NULL_COLLATION</td>
<td>SQL_NC_LOW (sorted at the low end of the list)</td>
</tr>
<tr>
<td>Description</td>
<td>InfoType Argument</td>
<td>Returns</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Numeric functions supported</td>
<td>SQL_NUMERIC_FUNCTIONS</td>
<td>SQL_FN_NUM_ABS</td>
</tr>
<tr>
<td>ODBC API conformance level</td>
<td>SQL_ODBC_API_CONFORMANCE</td>
<td>SQL_OAC_LEVEL1</td>
</tr>
<tr>
<td>SQL Access Group (SAG) conformance</td>
<td>SQL_ODBC_SAG_CLI_CONFIGURATION</td>
<td>SQL_OSCC_COMPLIANT</td>
</tr>
<tr>
<td>ODBC SQL syntax conformance</td>
<td>SQL_ODBC_SQL_CONFORMANCE</td>
<td>SQL_OSC_EXTENDED</td>
</tr>
<tr>
<td>Referential integrity syntax support</td>
<td>SQL_ODBC_SQL_OPT_IEF(renamed for ODBC 3.0.)</td>
<td>See SQL_INTEGRITY</td>
</tr>
<tr>
<td>ODBC version supported by driver manager</td>
<td>SQL_ODBC_VER</td>
<td>Returned by ODBC driver manager</td>
</tr>
<tr>
<td>Whether columns in ORDER BY clause must also be in select list</td>
<td>SQL_ORDER_BY_COLUMNS_IN_SELECT</td>
<td>N</td>
</tr>
<tr>
<td>Support for outer joins</td>
<td>SQL_OUTER_JOINS</td>
<td>Y (supports)</td>
</tr>
<tr>
<td>Term for entity that has owner privileges on objects</td>
<td>SQL_OWNER_TERM(Renamed for ODBC 3.0.)</td>
<td>See SQL_SCHEMA_TERM</td>
</tr>
<tr>
<td>Statements that support use of owner qualifiers</td>
<td>SQL_OWNER_USAGE(Replaced for ODBC 3.0.)</td>
<td>See SQL_SCHEMA_USAGE</td>
</tr>
<tr>
<td>Description</td>
<td>InfoType Argument</td>
<td>Returns</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Characteristics of row counts available in a parameterized execution</td>
<td>SQL_PARAM_ARRAY_ROW_COUNTS(ODBC 3.0)</td>
<td>SQL_PARC_NO_BATCH (only one row count available, which is the cumulative row count resulting from the execution of the statement for the entire array of parameters)</td>
</tr>
<tr>
<td>Characteristics of result sets available in a parameterized execution</td>
<td>SQL_PARAM_ARRAY_SELECTS(ODBC 3.0)</td>
<td>SQL_PAS_NO_SELECT</td>
</tr>
<tr>
<td>Supported operations in SQLSetPos</td>
<td>SQL_POS_OPERATION(Deprecated in ODBC 3.0.)</td>
<td>0 (does not support SQLSetPos)</td>
</tr>
<tr>
<td>Statements that support positioned operations</td>
<td>SQL_POSITIONED_STATEMENTS(Deprecated in ODBC 3.0.)</td>
<td>SQL_PS_POSITIONED_DELETE</td>
</tr>
<tr>
<td>Term for procedures</td>
<td>SQL_PROCEDURE_TERM</td>
<td>procedure</td>
</tr>
<tr>
<td>SQL procedures support</td>
<td>SQL_PROCEDURES</td>
<td>Y (supports SQL procedures)</td>
</tr>
<tr>
<td>Position of qualifier in a qualified table name</td>
<td>SQL_QUALIFIER_LOCATION(Renamed for ODBC 3.0.)</td>
<td>See SQL_CATALOG_LOCATION</td>
</tr>
<tr>
<td>Character used to separate table, column qualifiers</td>
<td>SQL_QUALIFIER_NAME_SEPARATOR(Renamed for ODBC 3.0.)</td>
<td>See SQL_CATALOG_NAME_SEPARATOR</td>
</tr>
<tr>
<td>Term for object that qualifies table names</td>
<td>SQL_QUALIFIER_TERM(Renamed for ODBC 3.0.)</td>
<td>See SQL_CATALOG_TERM</td>
</tr>
<tr>
<td>Statements that support qualifiers</td>
<td>SQL_QUALIFIER_USAGE(Renamed for ODBC 3.0.)</td>
<td>See SQL_CATALOG_USAGE</td>
</tr>
<tr>
<td>Case-sensitivity of quoted user-supplied names</td>
<td>SQL_QUOTED_IDENTIFIER_CASE</td>
<td>SQL_IC_SENSITIVE (Quoted identifiers in SQL are case-sensitive and are stored in mixed case in the system catalog). SQL_IC_LOWER (Quoted identifiers in SQL are not case-sensitive and are stored in lowercase in the system catalog).</td>
</tr>
<tr>
<td>Detect changes to any row in mixed-cursor operations</td>
<td>SQL_ROW_UPDATES</td>
<td>N</td>
</tr>
<tr>
<td>Term for entity that has owner privileges on objects</td>
<td>SQL_SCHEMA_TERM(Renamed for ODBC 3.0. Was SQL_OWNER_TERM)</td>
<td>owner</td>
</tr>
<tr>
<td>Statements that support use of owner qualifiers</td>
<td>SQL_SCHEMA_USAGE(Renamed for ODBC 3.0. Was SQL_OWNER_USAGE)</td>
<td>SQL_OU_DML_STATEMENTS</td>
</tr>
<tr>
<td>Description</td>
<td>InfoType Argument</td>
<td>Returns</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>--------------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Concurrency control options supported for scrollable cursors</td>
<td>SQL_SC_CURRENCYCY (Deprecated in ODBC 3.0.)</td>
<td>SQL_SCCO_READ_ONLY</td>
</tr>
<tr>
<td>Options supported for scrollable cursors</td>
<td>SQL_SC_CURRENTOPTIONS</td>
<td>SQL_SO_FORWARD_ONLY</td>
</tr>
<tr>
<td>Character to permit wildcard characters in search strings</td>
<td>SQL_SEARCH_PATTERN_ESCAPE</td>
<td>\ (backslash)</td>
</tr>
<tr>
<td>Name of the system where the c-treeSQL data source resides</td>
<td>SQL_SERVER_NAME</td>
<td>(string containing the name)</td>
</tr>
<tr>
<td>Special characters allowed in user-supplied names</td>
<td>SQL_SPECIAL_CHARACTERS</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>Level of SQL-92 support</td>
<td>SQL_SQL92_CONFORMANCE(ODBC 3.0)</td>
<td>SQL_SC_SQL92_ENTRY (entry level SQL-92 compliant)</td>
</tr>
<tr>
<td>Datetime scalar functions supported</td>
<td>SQL_SQL92_DATETIME_FUNCTIONS(ODBC 3.0)</td>
<td>SQL_SDF_CURRENT_DATE</td>
</tr>
<tr>
<td>Behavior of DELETE statement that refers to a foreign key</td>
<td>SQL_SQL92_FOREIGN_KEY_DELETE_RULE(ODBC 3.0)</td>
<td>0 (not supported)</td>
</tr>
<tr>
<td>Behavior of UPDATE statement that refers to a foreign key</td>
<td>SQL_SQL92_FOREIGN_KEY_UPDATE_RULE(ODBC 3.0)</td>
<td>0 (not supported)</td>
</tr>
<tr>
<td>GRANT statement clauses supported</td>
<td>SQL_SQL92_GRANT(ODBC 3.0)</td>
<td>SQL_SG_DELETE_TABLE</td>
</tr>
<tr>
<td>Numeric scalar functions supported</td>
<td>SQL_SQL92_NUMERIC_VALUE_FUNCTIONS(ODBC 3.0)</td>
<td>SQL_SNVF_CHAR_LENGTH</td>
</tr>
<tr>
<td>Predicates supported</td>
<td>SQL_SQL92_PREDICATES(ODBC 3.0)</td>
<td>SQL_EXISTS</td>
</tr>
<tr>
<td>Relational join operators supported</td>
<td>SQL_SQL92_RELATIONAL_JOIN_OPERATORS(ODBC 3.0)</td>
<td>0 (not supported)</td>
</tr>
<tr>
<td>Description</td>
<td>InfoType Argument</td>
<td>Returns</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>--------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>REVOKE statement clauses supported</td>
<td>SQL_SQL92_REVOKE(ODBC 3.0)</td>
<td>SQL_SR_GRANT_OPTION_FOR</td>
</tr>
<tr>
<td>Row value constructor expressions supported</td>
<td>SQL_SQL92_ROW_VALUE_CONSTRUCTOR(ODBC 3.0)</td>
<td>0 (does not support)</td>
</tr>
<tr>
<td>String scalar functions supported</td>
<td>SQL_SQL92_STRING_FUNCTIONS(ODBC 3.0)</td>
<td>SQL_SSF_CONVERT</td>
</tr>
<tr>
<td>Value expressions supported</td>
<td>SQL_SQL92_VALUE_EXPRESSIONS(ODBC 3.0)</td>
<td>SQL_SVE_COALESCE</td>
</tr>
<tr>
<td>CLI standards to which the driver conforms</td>
<td>SQL_STANDARD_CLI_CONFORMANCE(ODBC 3.0)</td>
<td>SQL_SCC_XOPEN_CLI_VERSION1 (conforms to X/Open CLI version 1)</td>
</tr>
<tr>
<td>Supported attributes of a static cursor: subset 1</td>
<td>SQL_STATIC_CURSOR_ATTRIBUTES1(ODBC 3.0)</td>
<td>SQL_CA1_NEXT (supports SQL_FETCH_NEXT argument to SQLFetchScroll for static cursors)</td>
</tr>
<tr>
<td>Supported attributes of a static cursor: subset 2</td>
<td>SQL_STATIC_CURSOR_ATTRIBUTES2(ODBC 3.0)</td>
<td>0 (does not support any attributes in the subset)</td>
</tr>
<tr>
<td>Whether static cursor changes are detectable</td>
<td>SQL_STATICSENSITIVITY(Deprecated in ODBC 3.0.)</td>
<td>0 (does not support static cursors)</td>
</tr>
<tr>
<td>String functions supported</td>
<td>SQL_STRING_FUNCTIONS</td>
<td>SQL_FN_STR_ASCII</td>
</tr>
<tr>
<td>Description</td>
<td>InfoType Argument</td>
<td>Returns</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>---------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Predicates that support subqueries</td>
<td>SQL_SUBQUERIES</td>
<td>SQL_SQ_CORRELATED_SUBQUERIES</td>
</tr>
<tr>
<td>System functions supported</td>
<td>SQL_SYSTEM_FUNCTIONS</td>
<td>SQL_FN_SYS_DBNAME</td>
</tr>
<tr>
<td>Term for tables</td>
<td>SQL_TABLE_TERM</td>
<td>Table</td>
</tr>
<tr>
<td>Timestamp intervals supported for TIMESTAMPADD function</td>
<td>SQL_TIMEDATE_ADD_INTERVALS</td>
<td>SQL_TSI_FRAC_SECOND</td>
</tr>
<tr>
<td>Timestamp intervals supported for TIMESTAMPDIFF function</td>
<td>SQL_TIMEDATE_DIFF_INTERVALS</td>
<td>SQL_TSI_FRAC_SECOND</td>
</tr>
</tbody>
</table>
### 4.4 Supported Environment, Connection, and Statement Attributes

The following table details the driver attributes that the c-treeACE ODBC Driver supports. Applications can set and retrieve supported driver attributes through the following routines:

- `SQLGetEnvAttr()` and `SQLSetEnvAttr()` for environment attributes
- `SQLGetConnectAttr()` and `SQLSetConnectAttr()` for connection attributes
- `SQLGetStmtAttr()` and `SQLSetStmtAttr()` for statement attributes

<table>
<thead>
<tr>
<th>Description</th>
<th>InfoType Argument</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date-time functions supported</td>
<td>SQL_TIMEDATE_FUNCTIONS</td>
<td>SQL_FN_TD_CURDATE</td>
</tr>
<tr>
<td>Support for DML, DDL within transactions</td>
<td>SQL_TXN_CAPABLE</td>
<td>SQL_TC_ALL (supports both DML and DDL)</td>
</tr>
<tr>
<td>Options for setting transaction isolation levels</td>
<td>SQL_TXN_ISOLATION_OPTIO N</td>
<td>SQL_TXN_READ_UNCOMMITTED</td>
</tr>
<tr>
<td>UNION support</td>
<td>SQL_UNION</td>
<td>SQL_U_UNION</td>
</tr>
<tr>
<td>Name of user connected to the data source</td>
<td>SQL_USER_NAME</td>
<td>(string containing the name)</td>
</tr>
<tr>
<td>A character string that indicates the year of publication of the X/Open specification with which the version of the ODBC Driver Manager fully complies.</td>
<td>SQL_XOPEN_CLI_YEAR</td>
<td>1995</td>
</tr>
</tbody>
</table>
# Supported Environment, Connection, and Statement Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ODBC Environment Attributes</strong></td>
<td></td>
</tr>
<tr>
<td>SQL_ATTR_CONNECTION_POOLING</td>
<td>No</td>
</tr>
<tr>
<td>SQL_ATTR_CP_MATCH</td>
<td>No</td>
</tr>
<tr>
<td>SQL_ATTR_ODBC_VER</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ATTR_OUTPUT_NTS</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>ODBC Connection Attributes</strong></td>
<td></td>
</tr>
<tr>
<td>SQL_ATTR_ACCESS_MODE</td>
<td>Yes (supports both read_only and read_write connection modes)</td>
</tr>
<tr>
<td>SQL_ATTR_ASYNC_ENABLE</td>
<td>Yes (supports only SQL_ASYNC_ENABLE_OFF, for other values returns warning &quot;Option value changed&quot; and sets to default)</td>
</tr>
<tr>
<td>SQL_ATTR_AUTO_IPD</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ATTR_AUTOCOMMIT</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ATTR_CONNECTION_TIMEOUT</td>
<td>Yes (supports only 0, for other values returns warning &quot;Option value changed&quot; and sets to default)</td>
</tr>
<tr>
<td>SQL_ATTR_CURRENT_CATALOG</td>
<td>No</td>
</tr>
<tr>
<td>SQL_ATTR_LOGIN_TIMEOUT</td>
<td>Yes The default value is 15 seconds.</td>
</tr>
<tr>
<td>SQL_ATTR_ODBC_CURSORS</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ATTR_PACKET_SIZE</td>
<td>Yes (supports only default, for other values returns warning &quot;Option value changed&quot; and sets to default)</td>
</tr>
<tr>
<td>SQL_ATTR_Quiet_MODE</td>
<td>Yes (supports only default, for other values returns warning &quot;Option value changed&quot; and sets to default)</td>
</tr>
<tr>
<td>SQL_ATTR_TRACE</td>
<td>No</td>
</tr>
<tr>
<td>SQL_ATTR_TRACEFILE</td>
<td>No</td>
</tr>
<tr>
<td>SQL_ATTR_TRANSLATE_DLL</td>
<td>No</td>
</tr>
<tr>
<td>SQL_ATTR_TRANSLATE_LIB</td>
<td>No</td>
</tr>
<tr>
<td>SQL_ATTR_TRANSLATE_OPTION</td>
<td>No</td>
</tr>
<tr>
<td>SQL_ATTR_TXN_Isolation</td>
<td>Yes</td>
</tr>
<tr>
<td>Attribute</td>
<td>Supported</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>ODBC Statement Attributes</td>
<td></td>
</tr>
<tr>
<td>SQL_ATTR_APP_PARAM_DESC</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ATTR_APP_ROW_DESC</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ATTR_ASYNC_ENABLE</td>
<td>Yes (supports only SQL_ASYNC_ENABLE_OFF, for other values returns warning “Option value changed” and sets to default)</td>
</tr>
<tr>
<td>SQL_ATTR_CONCURRENCY</td>
<td>Yes (supports only SQL_CONCUR_READ_ONLY, for other values returns warning “Option value changed” and sets to default)</td>
</tr>
<tr>
<td>SQL_ATTR_CURSOR_SCROLLABLE</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ATTR_CURSOR_TYPE</td>
<td>Yes (supports only SQL_FORWARD_ONLY, for other values returns warning “Option value changed” and sets to default)</td>
</tr>
<tr>
<td>SQL_ATTR_CURSOR_SENSITIVITY</td>
<td>Yes (supports only SQL_INSENSITIVE, for other values returns warning “Option value changed” and sets to default)</td>
</tr>
<tr>
<td>SQL_ATTR_ENABLE_AUTO_IPD</td>
<td>No</td>
</tr>
<tr>
<td>SQL_ATTR_FETCH_BOOKMARK_PTR</td>
<td>No</td>
</tr>
<tr>
<td>SQL_ATTR_IMP_PARAM_DESC</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ATTR_IMP_ROW_DESC</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ATTR_KEYSET_SIZE</td>
<td>No</td>
</tr>
<tr>
<td>SQL_ATTR_MAX_LENGTH</td>
<td>No</td>
</tr>
<tr>
<td>SQL_ATTR_MAX_ROWS</td>
<td>Yes (Supported both as statement attribute and connection attribute. If it is used as a connection attribute, it will override the statement attribute.)</td>
</tr>
<tr>
<td>SQL_ATTR_METADATA_ID</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ATTR_NOSCAN</td>
<td>No</td>
</tr>
<tr>
<td>SQL_ATTR_PARAM_BIND_OFFSET_PTR</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ATTR_PARAM_BIND_TYPE</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ATTR_PARAM_OPERATION_PTR</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ATTR_PARAM_STATUS_PTR</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL_ATTR_PARAMS_PROCESSED_PTR</td>
<td>Yes</td>
</tr>
</tbody>
</table>
4.5 Supported Functions

The c-treeSQL ODBC Driver supports all Core and Level 1 API functions, and most Level 2 functions. Here is a complete list of the functions the driver supports. Applications can request this same information through the `SQLGetFunctions()` function with a `FunctionId` of `SQL_API_ODBC3_ALL_FUNCTIONS`.

- `SQLAllocConnect()` (Deprecated in ODBC 3.0)
- `SQLAllocEnv()` (Deprecated in ODBC 3.0)
- `SQLAllocHandle()`
- `SQLAllocStmt()` (Deprecated in ODBC 3.0)
- `SQLBindCol()`
- `SQLBindParam()` (Deprecated in ODBC 3.0)
- `SQLBindParameter()`

### Attribute Conformance

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SQL_ATTR_PARAMSET_SIZE</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>SQL_ATTR_QUERY_TIMEOUT</code></td>
<td>Yes (defaults to 30 sec)</td>
</tr>
<tr>
<td><code>SQL_ATTR_RETRIEVE_DATA</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>SQL_ATTR_ROW_ARRAY_SIZE</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>SQL_ATTR_ROW_BIND_OFFSET_PTR</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>SQL_ATTR_ROW_BIND_TYPE</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>SQL_ATTR_ROW_NUMBER</code></td>
<td>No</td>
</tr>
<tr>
<td><code>SQL_ATTR_ROW_OPERATION_PTR</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>SQL_ATTR_ROW_STATUS_PTR</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>SQL_ATTR_ROWS_FETCHED_PTR</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>SQL_ATTR_SIMULATE_CURSOR</code></td>
<td>No</td>
</tr>
<tr>
<td><code>SQL_ATTR_USE_BOOKMARKS</code></td>
<td>No</td>
</tr>
</tbody>
</table>
Binds a buffer to a parameter marker in a SQL statement. **SQLBindParameter()** has 10 parameters. Parameter 4 and 5 change the way the date field is handled with reference to `TPE_DFLT_DATE`.

- **ValueType** [Input] The C date type of the parameter
- **ParameterType** [Input] The SQL data type of the parameter

When date is specified using a character string by setting **ValueType** to SQL_C_CHAR, the C data type of the parameter, the date formats supported are as follows:

When **ParameterType** is set to SQL_CHAR and **TPE_DFLT_DATE** is set to ISO_DFLT_DATE, the date formats supported is only YYYY-MM-DD and also the same format with ‘/’ as separator.

When **ParameterType** is set to SQL_TYPE_DATE and **TPE_DFLT_DATE** is set to ISO_DFLT_DATE, the date formats supported are MM-DD-YYYY, YYYY-MM-DD and DD-MON-YYYY and also the same formats with ‘/’ as the separator.

**SQLBrowseConnect()**

**SQLCancel()**

**SQLCloseCursor()**

**SQLColAttribute()**

**SQLColAttributes()**

(Deprecated in ODBC 3.0)

**SQLColumnPrivileges()**

**SQLColumnPrivileges()**

(Deprecated in ODBC 3.0)

**SQLColumns()**

**SQLConnect()**

**SQLCopyDesc()**

**SQLDataSources()**

**SQLDescribeCol()**

**SQLDescribeParam()**

**SQLDisconnect()**

**SQLDriverConnect()**

**SQLDrivers()**

**SQLEndTran()**

**SQLError()**

(Deprecated in ODBC 3.0)

**SQLExecDirect()**

**SQLExecute()**

**SQLExtendedFetch()**

(Deprecated in ODBC 3.0)
SQLFetch()
SQLFetchScroll()
SQLForeignKeys()
SQLFreeConnect()
  (Deprecated in ODBC 3.0)
SQLFreeEnv()
  (Deprecated in ODBC 3.0)
SQLFreeHandle()
SQLFreeStmt()
  (Deprecated in ODBC 3.0)
SQLGetConnectAttr()
SQLGetConnectOption()
  (Deprecated in ODBC 3.0)
SQLGetCursorName()
SQLGetData()
SQLGetDescField()
SQLGetDescRec()
SQLGetDiagField()
SQLGetDiagRec()
SQLGetEnvAttr()
SQLGetFunctions()
SQLGetInfo()
SQLGetStmtAttr()
SQLGetStmtOption()
  (Deprecated in ODBC 3.0)
SQLGetTypeInfo()
SQLMoreResults()
SQLNativeSqlW()
SQLNumParams()
SQLNumResultCols()
SQLParamData()
SQLParamOptions()
  (Deprecated in ODBC 3.0)
SQLPrepare()
SQLPrimaryKeys()
4.6 Supported Data Types

c-treeACE SQL supports a set of data types that the c-treeACE ODBC Driver maps to corresponding ODBC SQL data types. The following table lists the c-treeACE SQL data types and the corresponding ODBC data types.

c-treeSQL ODBC and Corresponding ODBC Data Types

<table>
<thead>
<tr>
<th>c-treeACE ODBC Data Type</th>
<th>ODBC Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT</td>
<td>SQL_BIGINT</td>
</tr>
<tr>
<td>BINARY</td>
<td>SQL_BINARY</td>
</tr>
<tr>
<td>BIT</td>
<td>SQL_BIT</td>
</tr>
<tr>
<td>CHAR or CHARACTER</td>
<td>SQL_CHAR</td>
</tr>
</tbody>
</table>
### 4.7 Thread Safety of c-treeACE ODBC Driver

By default, the c-treeACE ODBC Driver is thread-safe. However, this may not always be desired as it involves overhead on the performance of the system. Single threaded ODBC applications do not require thread safety as only one thread is involved.

To disable the thread safety feature, the following runtime flag is used:

```
DH_DISABLE_ODBC_THREAD_SAFETY
```

This flag must be set in `dhsq.ini` for c-treeACE SQL on Windows NT. In Unix, it will be an environment variable in the setup file.

By default, the thread safety feature is enabled. To disable this feature, set:

```
DH_DISABLE_ODBC_THREAD_SAFETY = Y
```
### 4.8 Error Messages

General SQL error messages are documented in “Error Messages” the *c-treeACE SQL Reference Manual*.

Aside from the error messages documented in the said manual, there is another known error that may occur when using c-treeACE ODBC Driver with Crystal Reports. This error may occur when generating reports that require a left outer join between tables. The cause of this error is due to the syntax used by Crystal Reports when creating the SQL statement. Users can specify the proper syntax to use by changing a key in the Windows registry. For more information about how to change the registry, please refer to this document: *How to Resolve the Join Syntax Errors in Crystal Reports 9* (http://scn.sap.com/docs/DOC-21392). Or search the SAP Community Network (http://scn.sap.com/welcome) for: Crystal left outer registry.

**Driver Specific Errors**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>Success</td>
</tr>
<tr>
<td>01000</td>
<td>General Warning</td>
</tr>
<tr>
<td>01000</td>
<td>Error while freeing stmt</td>
</tr>
<tr>
<td>01001</td>
<td>Cursor operation conflict</td>
</tr>
<tr>
<td>01002</td>
<td>Disconnect Error</td>
</tr>
<tr>
<td>01003</td>
<td>NULL value eliminated in set function</td>
</tr>
<tr>
<td>01004</td>
<td>String data, right truncated</td>
</tr>
<tr>
<td>01006</td>
<td>Privilege not revoked</td>
</tr>
<tr>
<td>01007</td>
<td>Privilege not granted</td>
</tr>
<tr>
<td>01S00</td>
<td>Invalid connection string attribute</td>
</tr>
<tr>
<td>01S01</td>
<td>Error in row</td>
</tr>
<tr>
<td>01S02</td>
<td>Option value changed</td>
</tr>
<tr>
<td>01S06</td>
<td>Attempt to fetch before the result set returned the first result set</td>
</tr>
<tr>
<td>01S07</td>
<td>Fractional Truncation</td>
</tr>
<tr>
<td>01S08</td>
<td>Error saving File DSN</td>
</tr>
<tr>
<td>01S09</td>
<td>Invalid keyword</td>
</tr>
<tr>
<td>07001</td>
<td>Wrong number of parameters</td>
</tr>
<tr>
<td>07002</td>
<td>COUNT field incorrect</td>
</tr>
<tr>
<td>07005</td>
<td>Prepared statement not a cursor specification</td>
</tr>
<tr>
<td>07006</td>
<td>Restricted data type attribute violation</td>
</tr>
<tr>
<td>07009</td>
<td>Invalid descriptor index</td>
</tr>
<tr>
<td>07S01</td>
<td>Invalid use of default parameter</td>
</tr>
<tr>
<td>08001</td>
<td>Client unable to establish connection</td>
</tr>
<tr>
<td>08002</td>
<td>Connection name in use</td>
</tr>
<tr>
<td>08003</td>
<td>Connection does not exist</td>
</tr>
<tr>
<td>08004</td>
<td>Server rejected the connection</td>
</tr>
<tr>
<td>Error Code</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>08007</td>
<td>Connection failure during transaction</td>
</tr>
<tr>
<td>08S01</td>
<td>Communication link failure</td>
</tr>
<tr>
<td>21S01</td>
<td>Insert value list does not match column list</td>
</tr>
<tr>
<td>21S02</td>
<td>Degree of derived table does not match column list</td>
</tr>
<tr>
<td>22001</td>
<td>String data right truncation</td>
</tr>
<tr>
<td>22002</td>
<td>Indicator variable required but not supplied</td>
</tr>
<tr>
<td>22003</td>
<td>Numeric value out of range</td>
</tr>
<tr>
<td>22007</td>
<td>Invalid datetime format</td>
</tr>
<tr>
<td>22008</td>
<td>Datetime field overflow</td>
</tr>
<tr>
<td>22012</td>
<td>Division by zero</td>
</tr>
<tr>
<td>22015</td>
<td>Interval field overflow</td>
</tr>
<tr>
<td>22018</td>
<td>Invalid character value for cast specification</td>
</tr>
<tr>
<td>22019</td>
<td>Invalid escape character</td>
</tr>
<tr>
<td>22025</td>
<td>Invalid escape sequence</td>
</tr>
<tr>
<td>22026</td>
<td>String data, length mismatch</td>
</tr>
<tr>
<td>23000</td>
<td>Integrity constraint violation</td>
</tr>
<tr>
<td>24000</td>
<td>Invalid cursor state</td>
</tr>
<tr>
<td>25000</td>
<td>Invalid transaction state</td>
</tr>
<tr>
<td>25S01</td>
<td>Transaction state</td>
</tr>
<tr>
<td>25S02</td>
<td>Transaction is still active</td>
</tr>
<tr>
<td>25S03</td>
<td>Transaction is rolled back</td>
</tr>
<tr>
<td>28000</td>
<td>Invalid authorization specification</td>
</tr>
<tr>
<td>34000</td>
<td>Invalid cursor name</td>
</tr>
<tr>
<td>3C000</td>
<td>Duplicate cursor name</td>
</tr>
<tr>
<td>3D000</td>
<td>Invalid catalog name</td>
</tr>
<tr>
<td>3F000</td>
<td>Invalid schema name</td>
</tr>
<tr>
<td>40001</td>
<td>Serialization failure</td>
</tr>
<tr>
<td>40003</td>
<td>Statement completion unknown</td>
</tr>
<tr>
<td>42000</td>
<td>Syntax error or access violation</td>
</tr>
<tr>
<td>42S01</td>
<td>Base Table or view already exists</td>
</tr>
<tr>
<td>42S02</td>
<td>Base Table or view not found</td>
</tr>
<tr>
<td>42S11</td>
<td>Index already exists</td>
</tr>
<tr>
<td>42S12</td>
<td>Index not found</td>
</tr>
<tr>
<td>42S21</td>
<td>Column already exists</td>
</tr>
<tr>
<td>42S22</td>
<td>Column not found</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------</td>
</tr>
<tr>
<td>44000</td>
<td>WITH CHECK OPTION violation</td>
</tr>
<tr>
<td>HY000</td>
<td>General Error</td>
</tr>
<tr>
<td>HY000</td>
<td>Operation invalid on a Read only connection</td>
</tr>
<tr>
<td>HY000</td>
<td>User cancelled operation</td>
</tr>
<tr>
<td>HY000</td>
<td>Host and Database must be specified</td>
</tr>
<tr>
<td>HY000</td>
<td>Invalid column number</td>
</tr>
<tr>
<td>HY000</td>
<td>Table type out of range</td>
</tr>
<tr>
<td>HY000</td>
<td>No data at execution values pending</td>
</tr>
<tr>
<td>HY000</td>
<td>Not supported for long data</td>
</tr>
<tr>
<td>HY000</td>
<td>Failed to Obtain Mutex Lock</td>
</tr>
<tr>
<td>HY000</td>
<td>Failed to Acquire Mutex Handle</td>
</tr>
<tr>
<td>HY001</td>
<td>Memory Allocation Failure</td>
</tr>
<tr>
<td>HY003</td>
<td>Invalid application buffer type</td>
</tr>
<tr>
<td>HY004</td>
<td>Invalid SQL data type</td>
</tr>
<tr>
<td>HY007</td>
<td>Associated statement is not prepared</td>
</tr>
<tr>
<td>HY008</td>
<td>Operation cancelled</td>
</tr>
<tr>
<td>HY009</td>
<td>Invalid use of null pointer</td>
</tr>
<tr>
<td>HY010</td>
<td>Function Sequence Error</td>
</tr>
<tr>
<td>HY011</td>
<td>Attribute cannot be set now</td>
</tr>
<tr>
<td>HY012</td>
<td>Invalid transaction operation code</td>
</tr>
<tr>
<td>HY013</td>
<td>Memory management error</td>
</tr>
<tr>
<td>HY014</td>
<td>Limit on the number of handles exceeded</td>
</tr>
<tr>
<td>HY015</td>
<td>No cursor name available</td>
</tr>
<tr>
<td>HY016</td>
<td>Cannot modify an implementation row descriptor</td>
</tr>
<tr>
<td>HY017</td>
<td>Invalid use of an automatically allocated descriptor handle</td>
</tr>
<tr>
<td>HY018</td>
<td>Server declined cancel request</td>
</tr>
<tr>
<td>HY019</td>
<td>Non-character and non-binary data sent in pieces</td>
</tr>
<tr>
<td>HY020</td>
<td>Attempt to concatenate a null value</td>
</tr>
<tr>
<td>HY021</td>
<td>Inconsistent Descriptor Info</td>
</tr>
<tr>
<td>HY024</td>
<td>Invalid attribute value</td>
</tr>
<tr>
<td>HY090</td>
<td>Invalid string or buffer length</td>
</tr>
<tr>
<td>HY091</td>
<td>Invalid descriptor field identifier</td>
</tr>
<tr>
<td>HY092</td>
<td>Invalid attribute /option identifier</td>
</tr>
<tr>
<td>HY093</td>
<td>Invalid parameter number</td>
</tr>
<tr>
<td>HY095</td>
<td>Function type out of range</td>
</tr>
<tr>
<td>HY096</td>
<td>Invalid information type</td>
</tr>
<tr>
<td>HY097</td>
<td>Column type out of range</td>
</tr>
<tr>
<td>HY098</td>
<td>Scope type out of range</td>
</tr>
<tr>
<td>HY099</td>
<td>Nullable type out of range</td>
</tr>
<tr>
<td>HY100</td>
<td>Uniqueness option type out of range</td>
</tr>
<tr>
<td>HY101</td>
<td>Accuracy option type out of range</td>
</tr>
<tr>
<td>HY103</td>
<td>Invalid retrieval code</td>
</tr>
<tr>
<td>HY104</td>
<td>Invalid precision or scale value</td>
</tr>
<tr>
<td>HY105</td>
<td>Invalid parameter type</td>
</tr>
<tr>
<td>HY106</td>
<td>Fetch type out of range</td>
</tr>
<tr>
<td>HY107</td>
<td>Row value out of range</td>
</tr>
<tr>
<td>HY109</td>
<td>Invalid cursor position</td>
</tr>
<tr>
<td>HY110</td>
<td>Invalid driver completion</td>
</tr>
<tr>
<td>HY111</td>
<td>Invalid bookmark value</td>
</tr>
<tr>
<td>HYC00</td>
<td>Optional feature not implemented</td>
</tr>
<tr>
<td>HYT00</td>
<td>Timeout expired</td>
</tr>
<tr>
<td>HYT01</td>
<td>Connection timeout expired</td>
</tr>
</tbody>
</table>

| IM001     | Driver does not support this function |
| IM002     | Data source not found and no default driver specified |
| IM003     | Specified driver could not be loaded |
| IM004     | Driver's SQLAllocHandle on SQL_HANDLE_ENV failed |
| IM005     | Driver's SQLAllocHandle on SQL_HANDLE_DBC failed |
| IM006     | Driver's SQLSetConnectAttr failed   |
| IM007     | No datasource or driver specified; dialog prohibited |
| IM008     | Dialog failed                      |
| IM009     | Unable to load translation DLL     |
| IM010     | Data source name too long          |
| IM011     | Driver name too long               |
| IM012     | DRIVER keyword syntax error        |
| IM013     | Trace file error                   |
| IM014     | Invalid name of File DSN           |
| IM015     | Corrupt file data source           |
| HY000     | Invalid Character Set Name         |
5. Direct Link c-treeACE SQL ODBC Driver

5.1 Introduction

The Open Database Connectivity (ODBC) interface from Microsoft has emerged as the standard mechanism for client applications to access data from a variety of different data sources through a single interface.

To become accessible from ODBC client applications, database environments must provide a software driver on the client system where the application resides. The driver translates the standard ODBC function calls into calls the database server can process, and returns the resulting data to the application.

The c-treeACE SQL ODBC Drivers are now available on Unix systems, including all Linux distributions supported by FairCom. The c-treeACE SQL ODBC implementation uses a method called “direct link” where the ODBC client application links directly to FairCom ODBC libraries, making unnecessary the use of “middleware” ODBC management software to manage ODBC connections and data sources.

In addition, this direct link driver can also be used by Microsoft Windows applications. The direct link approach avoids the necessity of dealing with the ODBC Manager when installing ODBC applications using FairCom’s ODBC driver, which may be advantageous on large scale systems or mass distributed products.

5.2 Linking the c-treeACE SQL Direct Link Driver

ODBC client applications that use the c-treeACE SQL ODBC direct link driver must include the following header files in any source modules that make calls to the ODBC API:

```c
#include <sql.h>
#include <sqlext.h>
```

These files are found in the \FairCom\V10.0.0\<platform>\include directory

c-treeACE SQL ODBC client applications must link the following libraries to resolve any calls to the ODBC API:

- \FairCom\V10.0.0\<platform>\lib\sql.odbc\libodbc_c.a
- \FairCom\V10.0.0\<platform>\lib\sql.odbc\libctesql.a
5.3 Unix/Linux ODBC Managers

Rather than linking directly with the c-treeACE SQL direct link ODBC driver, an application may link with a generic ODBC manager library, which loads the proper ODBC driver at runtime based on a requested DSN. This is the standard ODBC behavior on MS Windows. Unix platforms support various driver managers. One of the most common managers is the open source unixODBC library. For details on other proprietary Unix managers you are referred to their specific documentation.

To configure the unixODBC environment set the ODBCINI environment variable to refer to an .ini file (for example, odbc.ini) with your DSN information in your application environment. If ODBCINI is not set or unable to be opened, the driver will search $HOME/.odbc.ini for connection information.

The following example shows the ODBCINI setting for path /usr/local/unixODBC/etc/:

```
ODBCINI=/usr/local/unixODBC/etc/odbc.ini;
setenv ODBCINI /usr/local/unixODBC/etc/odbc.ini
```

Remember to set LD_LIBRARY_PATH with the path of the unixODBC install.

**Note:** The example below shows the setting for the path /usr/local/unixODBC/lib and c-tree path /home/FairCom/linux.x64.64bit/lib/sql.odbc:

```
setenv LD_LIBRARY_PATH /usr/local/unixODBC/lib:/home/FairCom/linux.x64.64bit/lib/sql.odbc
```

Here are typical settings for the c-treeACE SQL Unix ODBC driver to connect to a default server configuration on the local machine.

**Note:** The example below assumes that c-treeACE is installed in /home/FairCom//linux.x64.64bit:

```
[ctreeSQL]
Driver  = /home/FairCom/linux.x64.64bit/lib/sql.odbc/libctodbc.so
Host = localhost
Database = ctreeSQL
User ID = ADMIN
Password = ADMIN
Service = 6597
```

Below is information specific to unixODBC. This may vary across installations.

When `SQLConnect()` is called, unixODBC looks in the location specific by $ODBCINI for a matching DSN, and loads the specified "Driver". The "Driver" value may refer directly to a shared object to load, or another key.

If it is not a shared object, then you need a reference to the "Driver" value wherever the driver manager looks. With unixODBC, it typically looks in /etc/odbcinst.ini.

```
[CtreeSQLDriver]
Description = ODBC for ctreeSQL
Driver      = home/FairCom/linux.x64.64bit/lib/sql.odbc/libctodbc.so
```

With unixODBC, tracing can be enabled by adding the following to /etc/odbcinst.ini:

```
[ODBC]
Trace = Yes
TraceFile = /tmp/mytrace.log
```

This will log all ODBC calls with their parameter values and the return values to the TracingFile.

c-treeACE SQL PHP Example using unixODBC

```php
<?php
$pass=0;
$fail=0;

putenv("ODBCINI=/home/fctech/php-5.3.3/sapi/cli/odbc.ini");

$isc = odbc_connect('fc','admin','ADMIN');
if($isc == FALSE)
{
    $fail++;
    echo "FAIL - odbc_connect\n";
} else
{
    $pass++;
    echo "PASS - odbc_connect\n";
}

$test = odbc_exec($isc,"SELECT TOP 1 owner,tbl,creator from systables");
if($test == FALSE)
{
    $fail++;
    echo "FAIL - odbc_exec\n";
} else
{
    $pass++;
    echo "PASS - odbc_exec\n";
}

$val = odbc_result_all($test);
if($val == FALSE)
{
    $fail++;
    echo "FAIL - odbc_result_all\n";
}
```
6. Migration to c-treeACE SQL ODBC

6.1 Migration From c-tree Plus ODBC to c-treeACE SQL ODBC

With more and more traditional c-tree users leveraging the power of c-treeACE SQL, FairCom has labored to keep this transition as easy as possible. While we strive for the utmost in compatibility, we occasionally find areas where it is possible to encounter unexpected situations. The c-tree ODBC and c-treeACE SQL ODBC drivers are both fully standards compliant, however, subtle differences do exist. We have summarized these differences here.

- All table names returned by the c-treeACE SQL ODBC Driver are as ‘owner.tablename’ without regard to the server configuration option SQL_OPTION OWNER_FILE_NAME.
- The SQL_OPTION OWNER_FILE_NAMES only affects the name of the table created on disk. It determines whether the owner name is pre-pended to the physical table name or not. Table names returned by the ODBC driver are not affected by this c-treeACE SQL option.
- Using the c-tree ODBC driver it was common practice to represent the symbolic table name in uppercase. You can use uppercase table names in c-treeACE SQL as long as you do not wrap them in double quotes.

Another option is to use the SQL_OPTION DB_CASE_INSENSITIVE keyword before creating the template database and the c-treeACE SQL databases.
7. Troubleshooting

7.1 c-treeACE setup error 126 during ODBC installation phase

Issue:
When you are installing your c-treeACE package you get an error 126 during the ODBC Installation Phase.

Note: This issue may happen on Windows Vista, Windows 7 and Windows 8.

Cause:
The needed Microsoft Visual C++ Redistributable package was not installed properly because of missing Windows dependencies.

Solutions:
Solution 1: Make sure your Windows operating system is up-to-date. Please use Windows Update to get all the latest updates.

Solution 2: Manually download and install the Visual C++ 2017 Redistributable Package from the Microsoft website.
8. **Glossary**

**add [an ODBC data source]**

Make a data source available to ODBC through the Add operation of the ODBC Administrator utility. Adding a data source tells ODBC where a specific database resides and which ODBC driver to use to access it. Adding a data source also invokes a setup dialog box for the particular driver so you can provide other details the driver needs to connect to the database.

**ADMIN**

The default owner name for all system tables in a c-treeACE SQL database. Users must qualify references to system tables as ADMIN.tablename.

**client**

Generally, in client/server systems, the part of the system that sends requests to servers and processes the results of those requests.

**c-treeACE SQL environment**

An open database created by the combination of c-treeACE SQL layered on top of the c-treeACE Server.

**data source**

See ODBC data source

**delete [an ODBC data source]**

Remove information about an ODBC data source through the Delete operation of the ODBC Administrator utility. Deleting a data source does not delete the database it corresponds to, but removes information about the database's location from the ODBC.INI file or registry key.

**delimited identifiers**

Names in SQL statements enclosed in double quotation marks (" "). Enclosing a name in double quotation marks preserves the case of the name and allows it to include reserved words and special characters. Subsequent references to a delimited identifier must also use enclosing double quotation marks.

**ODBC Administrator**

Microsoft-supplied utility to add and delete ODBC data sources and drivers. The Administrator maintains two files (or registry keys).

- **ODBC.INI** details all the ODBC data sources an application can access from a system, and specifies which driver ODBC should use to access each data source.
- **ODBCINST.INI** details all the drivers that have been installed on a system.

The installation procedure installs the Administrator if it is not already present on a system. To invoke it, click on the ODBC icon in the Control Panel applications of the Windows Program Manager.
**ODBC application**
Any program that calls ODBC functions and uses them to issue SQL statements. Many vendors have added ODBC capabilities to their existing Windows-based tools.

**ODBC data source**
In ODBC terminology, a specific combination of a database system, the operating system it uses, and any network software required to access it. Before applications can access a database through ODBC, you use the ODBC Administrator to add a data source -- register information about the database and an ODBC driver that can connect to it -- for that database. More than one data source name can refer to the same database, and deleting a data source does not delete the associated database.

**ODBC driver**
Vendor-supplied software that processes ODBC function calls for a specific data source. The driver connects to the data source, translates the standard SQL statements into syntax the data source can process, and returns data to the application.

**ODBC driver manager**
A Microsoft-supplied program that routes calls from an application to the appropriate ODBC driver for a data source.

**query expression**
The fundamental element in SQL syntax. Query expressions specify a result table derived from some combination of rows from the tables or views identified in the FROM clause of the expression. Query expressions are the basis of SELECT, DECLARE CURSOR, CREATE VIEW, and INSERT statements.

**result set**
Another term for result table.

**result table**
A temporary table of values derived from columns and rows of one or more tables that meet conditions specified by a query expression.

**SQL engine**
The core internal component of c-treeACE SQL. The SQL engine receives requests from the c-treeACE SQL ODBC driver, processes them, and returns results.

**view**
A virtual table that recreates the result table specified by a SELECT statement. No data is stored in a view, but other queries can refer to it as if it were a table containing data corresponding to the result table it specifies.
Before you begin using this guide, be sure to review the relevant terms and typographical conventions used in the documentation.

The following formatted items identify special information.

<table>
<thead>
<tr>
<th>Formatting convention</th>
<th>Type of Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold</strong></td>
<td>Used to emphasize a point or for variable expressions such as parameters</td>
</tr>
<tr>
<td><strong>CAPITALS</strong></td>
<td>Names of keys on the keyboard. For example, SHIFT, CTRL, or ALT+F4</td>
</tr>
<tr>
<td><em>FairCom Terminology</em></td>
<td><strong>FunctionName()</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Parameter</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Code Example</strong></td>
</tr>
<tr>
<td></td>
<td><strong>utility</strong></td>
</tr>
<tr>
<td></td>
<td><strong>filename</strong></td>
</tr>
<tr>
<td><strong>CONFIGURATION KEYWORD</strong></td>
<td><strong>CTREE_ERR</strong></td>
</tr>
<tr>
<td></td>
<td><strong>CTREE_ERR</strong></td>
</tr>
</tbody>
</table>
Index

T
Thread Safety of c-treeACE ODBC Driver ..........75
Transaction Processing ..................................32
Troubleshooting ............................................84

U
Unix/Linux ODBC Managers ...........................81
unixODBC ..................................................81
Using Reserved Keywords with Microsoft Excel ....50

W
Windows Registry Entries ...............................48