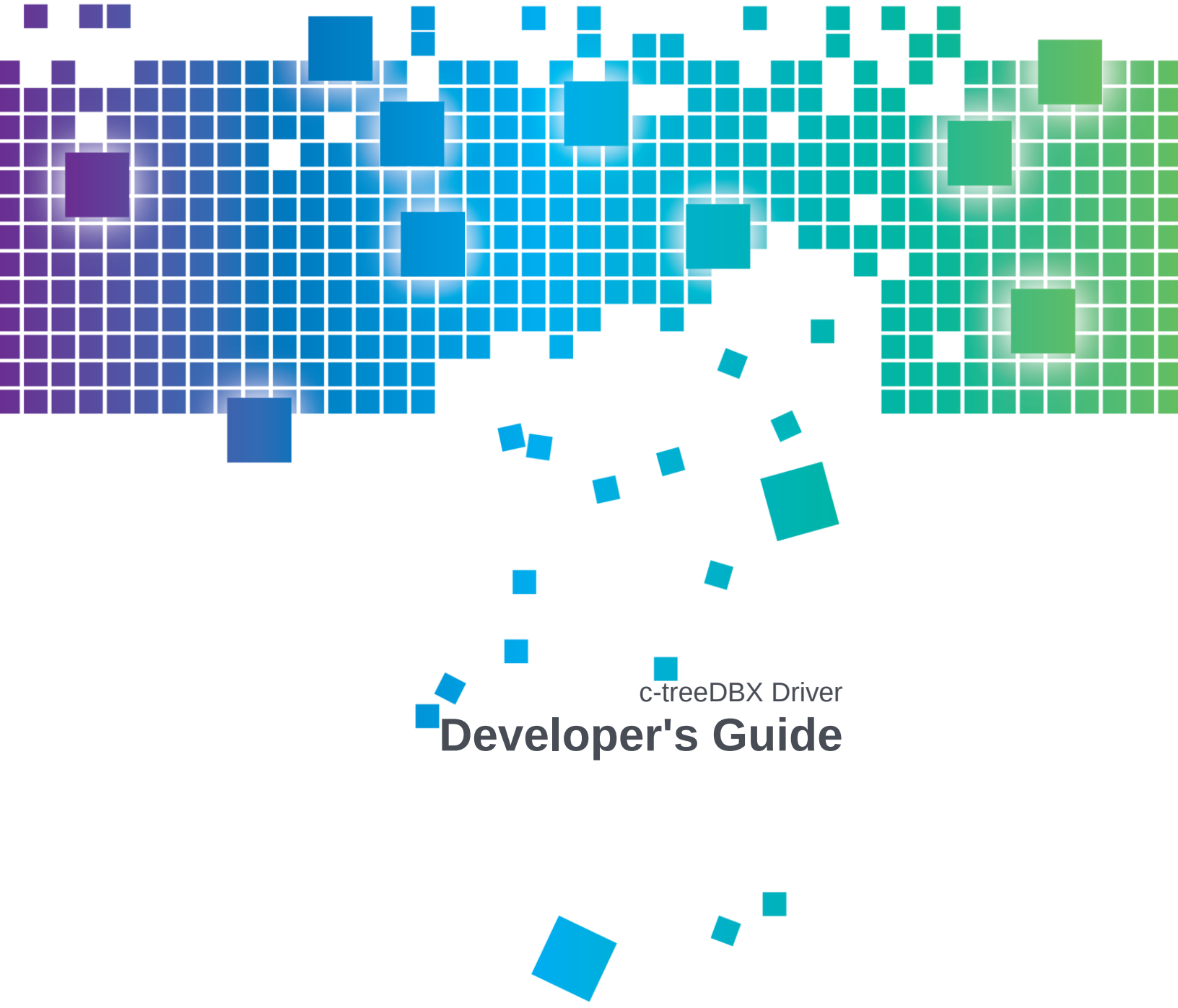
c-treeACE®
Advanced Core Engine



c-treeDBX Driver

Developer's Guide

Copyright Notice

Trademarks

11/16/2018

Contents

FairCom Typographical Conventions

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.

Formatting convention	Type of Information
Bold	Used to emphasize a point or for variable expressions such as parameters
CAPITALS	Names of keys on the keyboard. For example, SHIFT, CTRL, or ALT+F4
FairCom Terminology	FairCom technology term
FunctionName()	c-treeACE Function name
Parameter	c-treeACE Function Parameter
`Code Example`	Code example or Command line usage
utility	c-treeACE executable or utility
filename	c-treeACE file or path name
`CONFIGURATION KEYWORD`	c-treeACE Configuration Keyword
CTREE_ERR	c-treeACE Error Code

1. c-treeACE SQL dbExpress

Embarcadero's dbExpress provides developers superb database connectivity from their applications. dbExpress is a set of lightweight database drivers providing fast access to SQL database servers. When you deploy a c-treeACE SQL database application utilizing dbExpress, you need only include the c-treeACE SQL dbExpress Driver dll with the application files you build.

The c-treeACE SQL DBX Interface technology provides the driver necessary to connect to the c-treeACE SQL database engine. Quickly build client applications using the advanced RAD Studio development environment and efficiently access your c-treeACE SQL data through either Delphi or C++Builder.

By no means does this introduction cover the full scope, detail, or flexibility that the dbExpress technology, and c-treeDBX in particular, offers. It does, however, provide a quick glimpse into the advantages of using powerful c-treeACE SQL database technology in combination with a RAD tool. This is one of the easiest and quickest ways for your client application to access c-treeACE SQL. To learn more about dbExpress technology, visit embarcadero's web site www.embarcadero.com http://www.embarcadero.com/ or consult the help files of your RAD Studio tools.

1.1 Install c-treeDBX

RAD Studio Component Installation

You will need to be sure you properly register the dbExpress components in your RAD Studio environment. The c-treeACE dbExpress components will be copied to your specified directory location by the installer. However, the components will not be registered into the compiler(s) version(s) you have selected.

Install them through the **Component > Install Packages...** option in your Embarcadero RAD Studio XE package.

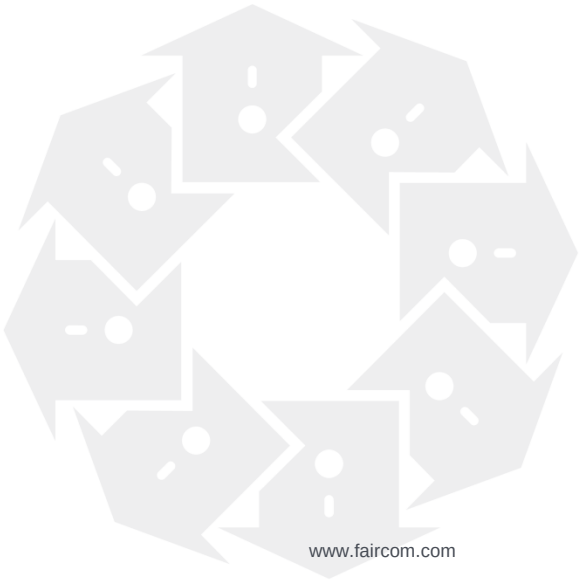
The packages are located in the following directories of your c-treeACE installation:

- **VCL for Delphi** - *C:\FairCom\V*\win32\sdk\ctree.vcl.delphi\src\<studio version>\ct<driver version>s<studio version>dt.bpl*
- **VCL for C++Builder** - *C:\FairCom\V*\win32\sdk\ctree.vcl.cbuilder\src\<studio version>\ct<driver version>s<studio version>dt.bpl*
- **dbExpress 4.0** - *C:\FairCom\V*\win32\sdk\sql.dbx\4.0\<studio version>\DBXCtreeDriver40<studio version>.bpl*

(Where *C:\FairCom\V** is the default installation directory.)

2. Quick Tour

This section introduces c-treeACE SQL dbExpress using a hands-on approach. The tutorial includes a sample project for developing a simple database application in RAD Studio. The full source is included with explanations of each part. The RAD Studio project files are included.



2.1 Introductory Tutorial

..\sdk\sql.dbx\Tutorial1\Unit1.pas

This tutorial will take you through the basic use of the c-treeACE SQL DBX Component - Delphi.

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:

```
procedure TForm1.btnStartClick(Sender: TObject);
begin
    Init;
    Define;
    Manage;
end;

procedure TForm1.btnDoneClick(Sender: TObject);
begin
    Done;
    Close;
end;
```

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()**:

```
procedure TForm1.Init;
var
    ininame : string;
    reg : TRegistry;
    ini : TIniFile;
begin
    // retrieve the driver ini file
    reg := TRegistry.Create;
    try
        Reg.RootKey := HKEY_CURRENT_USER;
        if Reg.OpenKey('\Software\Borland\DBExpress', True) then
        begin
            ininame := Reg.ReadString('Driver Registry File');
            Reg.CloseKey;
        end;
    finally
        reg.free;
    end;

    // retrieve the LibraryName and VendorLib from ini file
    ini := TIniFile.Create(ininame);
    try
        SQLConnection1.LibraryName := ini.ReadString('ctreeDBX', 'LibraryName',
'ctreedbx.dll');
        SQLConnection1.VendorLib := ini.ReadString('ctreeDBX', 'VendorLib', 'ctesql.dll');
    finally
        ini.free;
    end;

    // connect to server
    SimpleDataSet1.Active := false;
    SQLConnection1.Connected := true;
end;
```

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()**:

```
procedure TForm1.Define;
begin
    try
        // start transaction
        SQLConnection1.StartTransaction(td);
        // create table
        SQLConnection1.ExecuteDirect('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))');
        // commit transaction
        SQLConnection1.Commit(td);

    except on E: Exception do Handle_Exception(E);
    end;
end;
```

Manage

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

Below is the code for **Manage()**:

```
procedure TForm1.Manage;
begin
    // delete any existing records
    Delete_Records;
    // populate the table with data
    Add_Records;
    // display contents of table
    Display_Records;
end;



procedure TForm1.Delete_Records;
begin
    try
        SQLConnection1.StartTransaction(td);
        SQLConnection1.ExecuteDirect('DELETE FROM custmast');
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;


procedure TForm1.Add_Records;
const
    data : array [1..4] of string = (
        '(''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'')'
    );
    nRecords : integer = 4;
var
    i : integer;
    str : string;
begin
    try
        SQLConnection1.StartTransaction(td);
        for i := 1 to nRecords do
        begin
            // add one record at time to table
            str := 'INSERT INTO custmast VALUES ' + data[i];
            SQLConnection1.ExecuteDirect(str);
        end;
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
```

```
    end;
end;


procedure TForm1.Display_Records;
begin
    try
        SimpleDataSet1.DataSet.CommandType := ctQuery;
        SimpleDataSet1.DataSet.CommandText := 'SELECT * FROM custmast';
        SimpleDataSet1.Active := true;
    except on E: Exception do Handle_Exception(E);
    end;
end;
```

Done

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()**:

```
procedure TForm1.Done;
begin
    SimpleDataSet1.Active := false;
    SQLConnection1.Connected := false;
end;
```

Additional Resources

We encourage you to explore the additional resources listed here:

- Complete source code for this tutorial can be found in Unit1.pas in your installation directory, within the 'sdk\sql.dbx\Tutorial1' directory for your platform.
  Example for the Windows platform: C:\FairCom\V*\win32\sdk\sql.dbx\Tutorial1\Unit1.pas.
- Additional documentation may be found on the FairCom Web site at: www.faircom.com

2.2 Relationships

..\sdk\sql.dbx\Tutorial2\Unit1.pas

Now we will build some table/file relationships using the c-treeACE SQL DBX Component - Delphi.

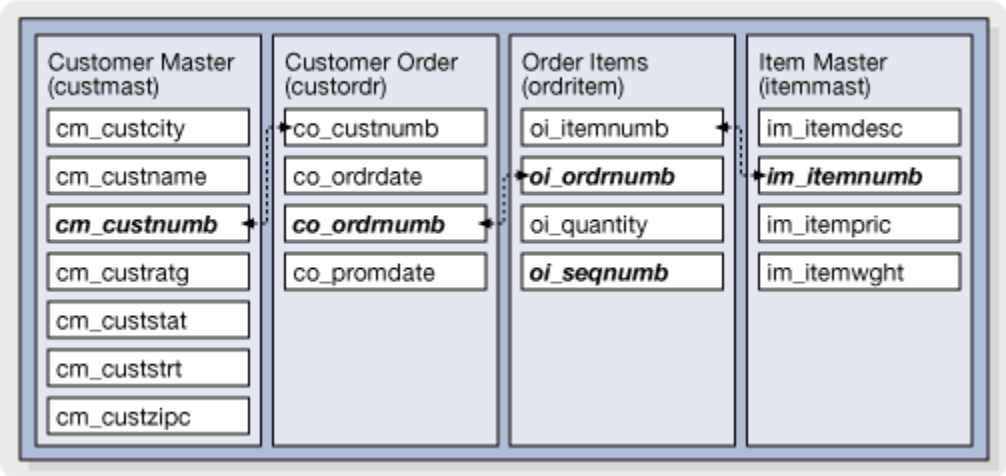
This tutorial will advance the concepts introduced in the first tutorial by expanding the number of tables. We will define key columns/fields and create specific indices for each table to form a relational model database.

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:

```
procedure TForm1.FormShow(Sender: TObject);
begin
    Initialize;
    Define;
    Manage;
end;

procedure TForm1.btnCloseClick(Sender: TObject);
```

```
begin
    Done;
    Close;
end;
```

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()**:

```
procedure TForm1.Initialize;
var
    ininame : string;
    reg : TRegistry;
    ini : TIniFile;
begin
    // retrieve the driver ini file
    reg := TRegistry.Create;
    try
        Reg.RootKey := HKEY_CURRENT_USER;
        if Reg.OpenKey('\Software\Borland\DBExpress', True) then
        begin
            ininame := Reg.ReadString('Driver Registry File');
            Reg.CloseKey;
        end;
    finally
        reg.free;
    end;
    // retrieve the LibraryName and VendorLib from ini file
    ini := TIniFile.Create(ininame);
    try
        SQLConnection1.LibraryName := ini.ReadString('ctreeDBX', 'LibraryName',
'ctreedbx.dll');
        SQLConnection1.VendorLib := ini.ReadString('ctreeDBX', 'VendorLib', 'ctesql.dll');
    finally
        ini.free;
    end;
    // establish the connection
    SQLConnection1.Connected := true;

end;
```

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()**:

```
procedure TForm1.Define();
begin
    // open or create the tables
    Create_CustomerMaster_Table;
    Create_CustomerOrders_Table;
    Create_OrderItems_Table;
    Create_ItemMaster_Table;
end;


procedure TForm1.Create_CustomerMaster_Table;
begin
    try
        SQLConnection1.StartTransaction(td);
        // create table
        SQLConnection1.ExecuteDirect('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))');
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;

    try
        SQLConnection1.StartTransaction(td);
        SQLConnection1.ExecuteDirect('CREATE UNIQUE INDEX cm_custnumb_idx ON custmast
(cm_custnumb)');
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;


procedure TForm1.Create_CustomerOrders_Table;
begin
    try
        SQLConnection1.StartTransaction(td);
        // create table
        SQLConnection1.ExecuteDirect('CREATE TABLE custordr (' +
            'co_ordrdate DATE,' +
            'co_promdate DATE,' +
            'co_ordrnumb CHAR(6),' +
            'co_custnumb CHAR(4))');
```

```
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;

    try
        SQLConnection1.StartTransaction(td);
        SQLConnection1.ExecuteDirect('CREATE UNIQUE INDEX co_ordrnumb_idx ON custordr
(co_ordrnumb)');
        SQLConnection1.ExecuteDirect('CREATE INDEX co_custnumb_idx ON custordr (co_custnumb)');
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;


procedure TForm1.Create_OrderItems_Table;
begin
    try
        SQLConnection1.StartTransaction(td);
        // create table
        SQLConnection1.ExecuteDirect('CREATE TABLE ordritem (' +
            'oi_sequnumb SMALLINT,' +
            'oi_quantity SMALLINT,' +
            'oi_ordrnumb CHAR(6),' +
            'oi_itemnumb CHAR(5))');
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;

    try
        SQLConnection1.StartTransaction(td);
        SQLConnection1.ExecuteDirect('CREATE UNIQUE INDEX oi_ordrnumb_idx ON ordritem
(oi_ordrnumb, oi_sequnumb)');
        SQLConnection1.ExecuteDirect('CREATE INDEX oi_itemnumb_idx ON ordritem (oi_itemnumb)');
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;

procedure TForm1.Create_ItemMaster_Table;
begin
    try
        SQLConnection1.StartTransaction(td);
        // create table
        SQLConnection1.ExecuteDirect('CREATE TABLE itemmast (' +
            'im_itemwght INTEGER,' +
            'im_itempric MONEY,' +
            'im_itemnumb CHAR(5),' +
            'im_itemdesc VARCHAR(47))');
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;

    try
        SQLConnection1.StartTransaction(td);
        SQLConnection1.ExecuteDirect('CREATE INDEX im_itemnumb_idx ON itemmast (im_itemnumb)');
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;
```

Manage

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

Below is the code for **Manage()**:

```
procedure TForm1.Manage();
var
  str : string;
begin
    // 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

    // name               total
    // @@@@@@@@@@@@@      $xx.xx

    // for each order in the CustomerOrders table
    //    fetch order number
    //    fetch customer number
    //    fetch name from CustomerMaster table based on customer number
    //    for each order item in OrderItems table
    //       fetch item quantity
    //       fetch item number
    //       fetch item price from ItemMaster table based on item number
    //    next
    // next

    try
        SQLQuery1.SQL.Clear;
        str := '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';

        SQLQuery1.SQL.Add(str);
        SQLQuery1.Prepared := true;
        SQLQuery1.Open;
        ClientDataSet1.Active := true;
    except on E: Exception do Handle_Exception(E);
    end;

end;


procedure TForm1.Add_CustomerMaster_Records;
const
    data : array [1..4] of string = (
```

```
        '(''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'')'
    );
    nRecords : integer = 4;
var
    i : integer;
    str : string;
begin
    try
        SQLConnection1.StartTransaction(td);
        Delete_Records('custmast');
        for i := 1 to nRecords do
        begin
            // add one record at time to table
            str := 'INSERT INTO custmast VALUES ' + data[i];
            SQLConnection1.ExecuteDirect(str);
        end;
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;


procedure TForm1.Add_CustomerOrders_Records;
const
    data : array [1..2] of string = (
        '(''09/01/2002'', ''09/05/2002'', ''1'', ''1001'')',
        '(''09/02/2002'', ''09/06/2002'', ''2'', ''1002'')'
    );
    nRecords : integer = 2;
var
    i : integer;
    str : string;
begin
    try
        SQLConnection1.StartTransaction(td);
        Delete_Records('custordr');
        for i := 1 to nRecords do
        begin
            // add one record at time to table
            str := 'INSERT INTO custordr VALUES ' + data[i];
            SQLConnection1.ExecuteDirect(str);
        end;
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;


procedure TForm1.Add_OrderItems_Records;
const
    data : array [1..4] of string = (
        '(''1'', ''2'', ''1'', ''1'')',
        '(''2'', ''1'', ''1'', ''2'')',
        '(''3'', ''1'', ''1'', ''3'')',
        '(''1'', ''3'', ''2'', ''3'')'
    );
```

```
    nRecords : integer = 4;
var
    i : integer;
    str : string;
begin
    try
        SQLConnection1.StartTransaction(td);
        Delete_Records('ordritem');
        for i := 1 to nRecords do
        begin
            // add one record at time to table
            str := 'INSERT INTO ordritem VALUES ' + data[i];
            SQLConnection1.ExecuteDirect(str);
        end;
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;


procedure TForm1.Add_ItemMaster_Records;
const
    data : array [1..4] of string = (
        '(''10'', ''19.95'', ''1'', ''Hammer'')',
        '( ''3'',  ''9.99'', ''2'', ''Wrench'')',
        '( ''4'',  ''16.59'',''3'', ''Saw'')',
        '( ''1'',  ''3.98'', ''4'', ''Pliers'')'
    );
    nRecords : integer = 4;
var
    i : integer;
    str : string;
begin
    try
        SQLConnection1.StartTransaction(td);
        Delete_Records('itemmast');
        for i := 1 to nRecords do
        begin
            // add one record at time to table
            str := 'INSERT INTO itemmast VALUES ' + data[i];
            SQLConnection1.ExecuteDirect(str);
        end;
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;


procedure TForm1.Delete_Records(Table : string);
var
    str : string;
begin
    try
        str := 'DELETE FROM ' + Table;
        SQLConnection1.ExecuteDirect(str);
    except on E: Exception do Handle_Exception(E);
    end;
end;
```

Done

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()**:

```
procedure TForm1.Done();
begin
    SQLConnection1.Connected := false;
end;
```

Additional Resources

We encourage you to explore the additional resources listed here:

- Complete source code for this tutorial can be found in Unit1.pas in your installation directory, within the 'sdk\sql.dbx\Tutorial2' directory for your platform.
  Example for the Windows platform: C:\FairCom\V*\win32\sdk\sql.dbx\Tutorial2\Unit1.pas.
- Additional documentation may be found on the FairCom Web site at: www.faircom.com

2.3 Record/Row Locking

..\sdk\sql.dbx\Tutorial3\Unit1.pas

Now we will explore row/record locks using the c-treeACE SQL DBX Component - Delphi.

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:

```
procedure TForm1.FormShow(Sender: TObject);
begin
    Initialize;
    Define;
    Manage;
end;

procedure TForm1.BtnCloseClick(Sender: TObject);
begin
    Done;
    Close;
end;
```

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()**:

```
procedure TForm1.Initialize();
var
    ininame : string;
    reg : TRegistry;
    ini : TIniFile;
begin
    // retrieve the driver ini file
    reg := TRegistry.Create;
    try
        Reg.RootKey := HKEY_CURRENT_USER;
        if Reg.OpenKey('\Software\Borland\DBExpress', True) then
        begin
            ininame := Reg.ReadString('Driver Registry File');
            Reg.CloseKey;
        end;
    finally
        reg.free;
    end;

    // retrieve the LibraryName and VendorLib from ini file
    ini := TIniFile.Create(ininame);
    try
        SQLConnection1.LibraryName := ini.ReadString('ctreeDBX', 'LibraryName',
'ctreedbx.dll');
        SQLConnection1.VendorLib := ini.ReadString('ctreeDBX', 'VendorLib', 'ctesql.dll');
    finally
        ini.free;
    end;

    // connect to server
    SQLConnection1.Connected := true;
end;
```

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()**:

```
procedure TForm1.Define();
begin
    try
        SQLConnection1.StartTransaction(td);
        // create table
        SQLConnection1.ExecuteDirect('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))');
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;

    try
        SQLConnection1.StartTransaction(td);
        SQLConnection1.ExecuteDirect('CREATE UNIQUE INDEX cm_custnumb_idx ON custmast
(cm_custnumb)');
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;
```

Manage

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

Below is the code for **Manage()**:

```
procedure TForm1.Manage();
begin
    // delete any existing records
    Delete_Records();
    // populate the table with data
    Add_CustomerMaster_Records();
    // display contents of table
    Display_Records();
end;


procedure TForm1.btmUpdateClick(Sender: TObject);
begin
    // update a record under locking control
    Update_CustomerMaster_Record();
    // display again after update and effects of lock
    Display_Records();
end;


procedure TForm1.Add_CustomerMaster_Records;
const
    data : array [1..4] of string = (
        '(''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'')'
    );
    nRecords : integer = 4;
var
    i : integer;
    str : string;
begin
    try
        SQLConnection1.StartTransaction(td);
        for i := 1 to nRecords do
        begin
            // add one record at time to table
            str := 'INSERT INTO custmast VALUES ' + data[i];
            SQLConnection1.ExecuteDirect(str);
        end;
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;
```

```
procedure TForm1.Delete_Records;
begin
    try
        SQLConnection1.ExecuteDirect('DELETE FROM custmast');
    except on E: Exception do Handle_Exception(E);
    end;
end;

procedure TForm1.Display_Records;
begin
    try
        SimpleDataSet1.Active := false;
        SimpleDataSet1.DataSet.CommandType := ctQuery;
        SimpleDataSet1.DataSet.CommandText := 'SELECT * FROM custmast';
        SimpleDataSet1.Active := true;
    except on E: Exception do Handle_Exception(E);
    end;
end;

procedure TForm1.Update_CustomerMaster_Record;
var
  str : string;
begin
    try
        SQLConnection1.StartTransaction(td);
        str := 'UPDATE custmast SET cm_custname = ''KEYON DOOLING'' where cm_custnumb = ''1003''';
        SQLConnection1.ExecuteDirect(str);

        Application.MessageBox('Click OK to unlock', '', 0);

        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;
```

Done

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()**:

```
procedure TForm1.Done();
begin
    SQLConnection1.Connected := false;
end;
```

Additional Resources

We encourage you to explore the additional resources listed here:

- Complete source code for this tutorial can be found in Unit1.pas in your installation directory, within the 'sdk\sql.dbx\Tutorial3' directory for your platform.
  Example for the Windows platform: C:\FairCom\V*\win32\sdk\sql.dbx\Tutorial3\Unit1.pas.
- Additional documentation may be found on the FairCom Web site at: www.faircom.com

2.4 Transaction Processing

..\sdk\sql.dbx\Tutorial4\Unit1.pas

Now we will discuss transaction processing as it relates to the c-treeACE SQL DBX Component - Delphi.

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:

```
procedure TForm1.FormShow(Sender: TObject);
begin
    Initialize;
    Define;
    Manage;
end;

procedure TForm1.BtnCloseClick(Sender: TObject);
begin
    Done;
    Close;
end;
```

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()**:

```
procedure TForm1.Initialize();
var
    ininame : string;
    reg : TRegistry;
    ini : TIniFile;
begin
    // retrieve the driver ini file
    reg := TRegistry.Create;
    try
        Reg.RootKey := HKEY_CURRENT_USER;
        if Reg.OpenKey('\Software\Borland\DBExpress', True) then
        begin
            ininame := Reg.ReadString('Driver Registry File');
            Reg.CloseKey;
        end;
    finally
        reg.free;
    end;

    // retrieve the LibraryName and VendorLib from ini file
    ini := TIniFile.Create(ininame);
    try
        SQLConnection1.LibraryName := ini.ReadString('ctreeDBX', 'LibraryName',
'ctreedbx.dll');
        SQLConnection1.VendorLib := ini.ReadString('ctreeDBX', 'VendorLib', 'ctesql.dll');
    finally
        ini.free;
    end;

    // connect to server
    SQLConnection1.Connected := true;
end;
```

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()**:

```
procedure TForm1.Define();
begin
    // delete tables ...
    Delete_Tables();
    // ...and re-create them with constraints
    Create_CustomerMaster_Table;
    Create_CustomerOrders_Table;
    Create_OrderItems_Table;
    Create_ItemMaster_Table;
end;



procedure TForm1.Delete_Tables;
begin
    try
        SQLConnection1.StartTransaction(td);
    except on E: Exception do Handle_Exception(E);
    end;
    try
        SQLConnection1.ExecuteDirect('DROP TABLE ordritem');
    except on E: Exception do Handle_Exception(E);
    end;
    try
        SQLConnection1.ExecuteDirect('DROP TABLE custordr');
    except on E: Exception do Handle_Exception(E);
    end;
    try
        SQLConnection1.ExecuteDirect('DROP TABLE custmast');
    except on E: Exception do Handle_Exception(E);
    end;
    try
        SQLConnection1.ExecuteDirect('DROP TABLE itemmast');
    except on E: Exception do Handle_Exception(E);
    end;
    try
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;


procedure TForm1.Create_CustomerMaster_Table;
begin
    try
        SQLConnection1.StartTransaction(td);
        // create table
```

```
        SQLConnection1.ExecuteDirect('CREATE TABLE custmast (' +
            'cm_custnumb CHAR(4) PRIMARY KEY,' +
            'cm_custzipc CHAR(9),' +
            'cm_custstat CHAR(2),' +
            'cm_custrtng CHAR(1),' +
            'cm_custname VARCHAR(47),' +
            'cm_custaddr VARCHAR(47),' +
            'cm_custcity VARCHAR(47))');
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;


procedure TForm1.Create_CustomerOrders_Table;
begin
    try
        SQLConnection1.StartTransaction(td);
        // create table
        SQLConnection1.ExecuteDirect('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)');
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;

procedure TForm1.Create_OrderItems_Table;
begin
    try
        SQLConnection1.StartTransaction(td);
        // create table
        SQLConnection1.ExecuteDirect('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)');
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;

procedure TForm1.Create_ItemMaster_Table;
begin
    try
        SQLConnection1.StartTransaction(td);
        // create table
        SQLConnection1.ExecuteDirect('CREATE TABLE itemmast (' +
            'im_itemwght INTEGER,' +
            'im_itempric MONEY,' +
            'im_itemnumb CHAR(5) PRIMARY KEY,' +
            'im_itemdesc VARCHAR(47))');
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;
```

Manage

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

Below is the code for **Manage()**:

```
procedure TForm1.Manage();
begin
    // populate the tables with data
    Add_CustomerMaster_Records;
    Add_ItemMaster_Records;

    Add_Transactions();

    // display the orders and their items
    Display_CustomerOrders();
    Display_OrderItems();
end;


procedure TForm1.Add_CustomerMaster_Records;
const
    data : array [1..4] of string = (
        '(''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'')'
    );
    nRecords : integer = 4;
var
    i : integer;
    str : string;
begin
    try
        SQLConnection1.StartTransaction(td);
        for i := 1 to nRecords do
        begin
            // add one record at time to table
            str := 'INSERT INTO custmast VALUES ' + data[i];
            SQLConnection1.ExecuteDirect(str);
        end;
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;


procedure TForm1.Add_ItemMaster_Records;
const
    data : array [1..4] of string = (
        '(''10'', ''19.95'', ''1'', ''Hammer'')',
        '( ''3'',  ''9.99'', ''2'', ''Wrench'')',
```

```
        '( ''4'',  ''16.59'',''3'', ''Saw'')',
        '( ''1'',  ''3.98'', ''4'', ''Pliers'')'
    );
    nRecords : integer = 4;
var
    i : integer;
    str : string;
begin
    try
        SQLConnection1.StartTransaction(td);
        for i := 1 to nRecords do
        begin
            // add one record at time to table
            str := 'INSERT INTO itemmast VALUES ' + data[i];
            SQLConnection1.ExecuteDirect(str);
        end;
        SQLConnection1.Commit(td);
    except on E: Exception do Handle_Exception(E);
    end;
end;


procedure TForm1.Add_Transactions;
const
    orders : array [1..3, 1..4] of string = (
        ('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')
    );
    items : array [1..6, 1..4] of string = (
        ('1', '1', '2', '1'),
        ('1', '2', '1', '2'),
        ('2', '1', '1', '3'),
        ('2', '2', '3', '4'),
        ('3', '1', '2', '3'),
        ('3', '2', '2', '99') // bad item number
    );
    nOrders : integer = 3;
    nItems : integer = 6;
var
    i : integer;
    j : integer;
    str : string;
begin
    j := 1;
    // process orders
    for i := 1 to nOrders do
    begin
        try
            // start a transaction
            SQLConnection1.StartTransaction(td);
            str := 'INSERT INTO custordr VALUES (' +
                '''' + orders[i,1] + ''', ' +
                '''' + orders[i,2] + ''', ' +
                '''' + orders[i,3] + ''', ' +
                '''' + orders[i,4] + ''')';
            // add order record
            SQLConnection1.ExecuteDirect(str);
        except on E: Exception do Handle_Exception(E);
        end;
```

```
        // process order items
        while (CompareStr(items[j,1], orders[i,3]) = 0) do
        begin
            try
                str := 'INSERT INTO ordritem VALUES (' +
                    items[j,2] + ', ' +
                    items[j,3] + ', ' +
                    '''' + items[j,1] + ''', ' +
                    '''' + items[j,4] + ''')';
                // add order item record
                SQLConnection1.ExecuteDirect(str);
            except on E: Exception do Handle_Exception(E);
            end;

            // bump to next item
            j := j + 1;

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

        try
            // commit the transaction
            SQLConnection1.Commit(td);
        except on E: Exception do
            Handle_Exception(E);
        end;
    end;
end;

procedure TForm1.Display_CustomerOrders();
begin
    ClientDataSet1.Active := false;
    SQLQuery1.SQL.Clear;
    SQLQuery1.SQL.Add('SELECT * FROM custordr');
    SQLQuery1.Prepared := true;
    SQLQuery1.Open;
    ClientDataSet1.Active := true;
end;

procedure TForm1.Display_OrderItems();
begin
    ClientDataSet2.Active := false;
    SQLQuery2.SQL.Clear;
    SQLQuery2.SQL.Add('SELECT * FROM ordritem');
    SQLQuery2.Prepared := true;
    SQLQuery2.Open;
    ClientDataSet2.Active := true;
end;
```

Done

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()**:

```
procedure TForm1.Done();
begin
    Delete_Tables();
    SQLConnection1.Connected := false;
end;
```

Additional Resources

We encourage you to explore the additional resources listed here:

- Complete source code for this tutorial can be found in Unit1.pas in your installation directory, within the 'sdk\sql.dbx\Tutorial4' directory for your platform.
  Example for the Windows platform: C:\FairCom\V*\win32\sdk\sql.dbx\Tutorial4\Unit1.pas.
- Additional documentation may be found on the FairCom Web site at: www.faircom.com

3. Index