

BlackBerry Java Application

SQLite

Version: 5.0

Development Guide

Contents

1	SQLite overview.....	2
2	Locations of SQLite database files.....	3
3	Best practice: Using a SQLite database browser with the BlackBerry Smartphone Simulator.....	4
4	Reducing database size by using the vacuum command.....	5
5	Best practice: Optimizing SQLite database performance.....	6
6	Tasks that support SQLite development.....	7
	Simulate a media card.....	7
	Navigate to a SQLite database file.....	7
7	Code samples.....	8
	Create a SQLite database.....	8
	Code sample: Creating a SQLite database.....	9
	Code sample: Creating an encrypted SQLite database.....	10
	Code sample: Adding a schema to a SQLite database.....	11
	Code sample: Inserting table data.....	13
	Code sample: Creating a parameterized insert.....	14
	Code sample: Retrieving table data.....	16
	Code sample: Deleting table data.....	17
	Code sample: Updating table data.....	18
	Code sample: Creating a parameterized update.....	20
	Code sample: Deleting a SQLite database.....	21
	Code sample: Listing database tables.....	22
	Code sample: Using transactions.....	24
8	Glossary.....	26
9	Provide feedback.....	27
10	Document revision history.....	28
11	Legal notice.....	29

SQLite overview

1

SQLite® is an open source relational database library. It is designed to make efficient use of memory resources and it includes few features. As a result, it can be a good choice for embedded and wireless applications. BlackBerry® devices that run BlackBerry® Device Software version 5.0 or later have the SQLite library integrated into the operating system and virtual machine. The SQLite API enables you to develop applications that use the integrated SQLite database. It was introduced in BlackBerry® Java® Development Environment version 5.0.

The `net.rim.device.api.database` package includes classes that enable you to work with SQLite.

You can use the following approach to work with an existing SQLite database.

- Create a SQL statement by calling `Database.createStatement()`.
- Prepare the statement to run by calling `Statement.prepare()`. Performing this step is like compiling the statement.
- Run the statement by calling `Statement.getCursor()` if the statement might return results and `Statement.execute()` otherwise.
- If there are results, retrieve them by iterating over the returned `Database.Cursor` row by row.

To learn more about developing applications that use SQLite, see the SQLite documentation at <http://www.sqlite.org> and the API reference for the BlackBerry Java Development Environment.

Locations of SQLite database files

2

Each SQLite® database is stored in a single file. If you only specify the database name as the parameter value to `DatabaseFactory.create()`, the database file is created on the SD card of the device. The default location for the database file is `/SDCard/databases/<application_name>/`. The name of the application that creates the database is included in the path to avoid name collisions.

You can create database files in eMMC memory, on devices that support it, by specifying the corresponding file system path.

Type of memory	File system path
SD card	<code>/SDCard/</code>
eMMC device memory	<code>/store/</code>
default	<code>/SDCard/databases/<application_name></code>

Best practice: Using a SQLite database browser with the BlackBerry Smartphone Simulator

3

When you develop an application that uses a SQLite® database, you might find it useful to work in a development environment that enables you to execute a SQL statement and look at the resulting data changes in the database. A convenient way to establish a development environment like that is to store your database on the SD card during development. As you develop your application, you can run it on the BlackBerry® Smartphone Simulator to check newly-added functionality. The BlackBerry Smartphone Simulator stores files saved on the emulated SD card in a specified directory on your development computer.

You can use a SQLite database browser to inspect and manage the SQLite databases that are stored on the file system of your desktop. You can use the BlackBerry Smartphone Simulator to run your application and access the SQLite database used by your application from your desktop file system. You can use the SQLite database browser to navigate to the SQLite database that is stored in the directory that the BlackBerry Smartphone Simulator uses to store the files saved on the emulated SD card. Using the database browser to view your database as you change it programmatically can give you real-time feedback about the changes your application is making to the database.

You might store your database on an SD card. However, BlackBerry® Bold™ Series and BlackBerry® Storm™ Series devices support storing SQLite databases on internal device memory. If your application is designed to store your SQLite database on internal device memory, you should implement your application so that it is easy to modify the code to change the storage location of the database.

Reducing database size by using the vacuum command

4

A SQLite® database is stored in a single file. Some database operations, such as dropping a table or inserting and deleting data, cause the file to become fragmented. You can use the vacuum command to defragment and reduce the size of the file.

The vacuum command copies all of the information required to recreate a database into a temporary file in memory and uses that information to create a new database file. As the command creates the new database file, it eliminates free pages, makes table data contiguous, and reorganizes the database file structure.

If you run the vacuum command on a BlackBerry® device that does not have enough memory available to store the required temporary information, the command will fail with an out of memory error.

Best practice: Optimizing SQLite database performance

5

To optimize the performance of a SQLite® database on a BlackBerry® device, you must consider both the database design and how your application uses the SQLite API to interact with the database.

Consider the following guidelines:

- Store as little data as possible. SQLite caches frequently accessed database pages. By storing less data you can increase the probability that the SQLite library retrieves requested data more quickly from the cache rather than from the relatively slow flash memory.
- Use temporary tables. Do this only if you do not need the data to be available following a reset of the BlackBerry device.
- Prepare generic statements that use named variables. Execute the statements when they are required by iterating through the variable values, binding the values to the named variables in each iteration.
- Use explicit transactions. Otherwise, a transaction begins before each statement is executed and ends after the statement is executed. This is inefficient. It requires the opening, reopening, writing to, and closing of the journal file for each statement.
- Avoid subqueries. By default, the SQLite® library stores the subquery results in a temporary file.

Tasks that support SQLite development

6

Simulate a media card

To test SQLite® database applications on the BlackBerry® Smartphone Simulator, you might have to configure the simulator to emulate a media card. By default, database files are stored on a media card.

1. Create a folder on your computer to store emulation files for the media card.
2. On the **Simulate** menu, click **Change SD Card**.
3. Click **Add Directory**.
4. Navigate to and click the folder you created.
5. Click **OK**.
6. Click **Close**.

The simulator adds files to the folder and begins emulating a media card. If you reset the simulator, you must restart the simulation of the media card.

Navigate to a SQLite database file

To verify that you have successfully created a new SQLite® database, you can navigate to the location where you expect to find that file by using the explore functionality of the media application. Each SQLite database is stored in a single file.

1. In the media application, press the **Menu** key.
2. Click **Explore**.
3. Navigate to the folder that should contain the database file to verify that the database file is in that folder.

Code samples

7

Create a SQLite database

1. Import the required libraries.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.database.*;
import net.rim.device.api.io.*;
```

2. Create the framework for the application by extending the `UiApplication` class. This class represents your application. Provide a `main()` method for the new class. In the `main()` method, create an instance of the new class and invoke the `enterEventDispatcher()` method to enable the application to receive events. Provide a constructor for the new class. In the constructor, call the `pushScreen` method to display the custom screen for the application.

```
public class CreateDatabase extends UiApplication
{
    public static void main(String[] args)
    {
        CreateDatabase theApp = new CreateDatabase();
        theApp.enterEventDispatcher();
    }

    public CreateDatabase()
    {
        pushScreen(new CreateDatabaseScreen());
    }
}
```

3. Create the screen for the application by extending the `MainScreen` class. Provide a constructor for the new class. In the constructor, create the title for the screen by using a `LabelField` object and display it by invoking the `setTitle()` method. Invoke the `add()` method to display a text field on the screen.

```
class CreateDatabaseScreen extends MainScreen
{
    public CreateDatabaseScreen()
    {
        LabelField title = new LabelField("SQLite Create Database Sample",
        LabelField.ELLIPSIS |
        LabelField.USE_ALL_WIDTH);
        setTitle(title);
        add(new RichTextField("Creating a database called " +
```

```
        "MyTestDatabase.db on the SDCard."));
    }
}
```

4. Create a URI that represents the database file by invoking the static `create()` method of the `URI` class. Invoke the `create()` method of the `DatabaseFactory` class to create the database that corresponds to the URI that you created.

```
try
{
    URI myURI = URI.create("file:///SDCard/Databases/SQLite_Guide/" +
        "MyTestDatabase.db");
    DatabaseFactory.create(myURI);
}
catch ( Exception e )
{
    System.out.println( e.getMessage() );
    e.printStackTrace();
}
```

Creates a SQLite® database on the SD card of the BlackBerry® device. If you do not specify the full path to the database to create, it will be created in a folder named after your application.

After you finish:

You can use the explore functionality of the media application to check that the database was created.

Code sample: Creating a SQLite database

```
/*
 * CreateDatabase.java
 *
 * Research In Motion Limited proprietary and confidential
 * Copyright Research In Motion Limited, 2010
 */
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.database.*;
import net.rim.device.api.io.*;

public class CreateDatabase extends UiApplication
{
    public static void main(String[] args)
    {
        CreateDatabase theApp = new CreateDatabase();
        theApp.enterEventDispatcher();
    }
}
```

```

public CreateDatabase()
{
    pushScreen(new CreateDatabaseScreen());
}
}

class CreateDatabaseScreen extends MainScreen
{
    Database d;
    public CreateDatabaseScreen()
    {
        LabelField title = new LabelField("SQLite Create Database Sample",
                                           LabelField.ELLIPSIS |
                                           LabelField.USE_ALL_WIDTH);

        setTitle(title);
        add(new RichTextField("Creating a database called " +
                             "MyTestDatabase.db on the SDCard."));

        try
        {
            URI myURI = URI.create("file:///SDCard/Databases/SQLite_Guide/" +
                                   "MyTestDatabase.db");
            d = DatabaseFactory.create(myURI);
            d.close();
        }
        catch ( Exception e )
        {
            System.out.println( e.getMessage() );
            e.printStackTrace();
        }
    }
}
}

```

Code sample: Creating an encrypted SQLite database

```

/* CreateEncryptedDatabase.java
 *
 * Research In Motion Limited proprietary and confidential
 * Copyright Research In Motion Limited, 2010
 */
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.database.*;
import net.rim.device.api.io.*;

public class CreateEncryptedDatabase extends UiApplication
{
    public static void main(String[] args)

```

```

{
    CreateEncryptedDatabase theApp = new CreateEncryptedDatabase();
    theApp.enterEventDispatcher();
}

public CreateEncryptedDatabase()
{
    pushScreen(new CreateEncryptedDatabaseScreen());
}
}

class CreateEncryptedDatabaseScreen extends MainScreen
{
    Database d;
    public CreateEncryptedDatabaseScreen()
    {
        LabelField title = new LabelField("SQLite Create Encrypted Database Sample",
        LabelField.ELLIPSIS |
        LabelField.USE_ALL_WIDTH);
        setTitle(title);
        add(new RichTextField("Creating an encrypted database called " +
        "MyEncryptedDatabase.db on the SDCard."));
        try
        {
            URI myURI = URI.create("file:///SDCard/Databases/SQLite_Guide/" +
            "MyEncryptedDatabase.db");
            DatabaseSecurityOptions dbso = new DatabaseSecurityOptions(true);
            d = DatabaseFactory.create(myURI,dbso);
            d.close();
        }
        catch ( Exception e )
        {
            System.out.println( e.getMessage() );
            e.printStackTrace();
        }
    }
}
}

```

Code sample: Adding a schema to a SQLite database

```

/*
 * CreateDatabaseSchema.java
 *
 * Research In Motion Limited proprietary and confidential
 * Copyright Research In Motion Limited, 2010
 */
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;

```

```

import net.rim.device.api.ui.container.*;
import net.rim.device.api.database.*;
import net.rim.device.api.io.*;

public class CreateDatabaseSchema extends UiApplication
{
    public static void main(String[] args)
    {
        CreateDatabaseSchema theApp = new CreateDatabaseSchema();
        theApp.enterEventDispatcher();
    }

    public CreateDatabaseSchema()
    {
        pushScreen(new CreateDatabaseSchemaScreen());
    }
}

class CreateDatabaseSchemaScreen extends MainScreen
{
    Database d;
    public CreateDatabaseSchemaScreen()
    {
        LabelField title = new LabelField("SQLite Create " +
                                           "Database Schema Sample",
                                           LabelField.ELLIPSIS |
                                           LabelField.USE_ALL_WIDTH);

        setTitle(title);
        add(new RichTextField("Adding a table to a database called " +
                              "MyTestDatabase.db on the SDCard."));

        try
        {
            URI myURI = URI.create("/SDCard/Databases/SQLite_Guide/" +
                                   "MyTestDatabase.db");
            d = DatabaseFactory.open(myURI);
            Statement st = d.createStatement( "CREATE TABLE 'People' ( " +
                                             "'Name' TEXT, " +
                                             "'Age' INTEGER )" );

            st.prepare();
            st.execute();
            st.close();
            d.close();
        }
        catch ( Exception e )
        {
            System.out.println( e.getMessage() );
            e.printStackTrace();
        }
    }
}

```

```
}  
}
```

Code sample: Inserting table data

```
/*  
 * InsertData.java  
 *  
 * Research In Motion Limited proprietary and confidential  
 * Copyright Research In Motion Limited, 2010  
 */  
import net.rim.device.api.ui.*;  
import net.rim.device.api.ui.component.*;  
import net.rim.device.api.ui.container.*;  
import net.rim.device.api.database.*;  
import net.rim.device.api.io.*;  
  
public class InsertData extends UiApplication  
{  
    public static void main(String[] args)  
    {  
        InsertData theApp = new InsertData();  
        theApp.enterEventDispatcher();  
    }  
  
    public InsertData()  
    {  
        pushScreen(new InsertDataScreen());  
    }  
}  
  
class InsertDataScreen extends MainScreen  
{  
    Database d;  
    public InsertDataScreen()  
    {  
        LabelField title = new LabelField("SQLite Insert Data " +  
                                           "Schema Sample",  
                                           LabelField.ELLIPSIS |  
                                           LabelField.USE_ALL_WIDTH);  
  
        setTitle(title);  
        add(new RichTextField("Attempting to insert data into " +  
                              "MyTestDatabase.db on the SDCard."));  
  
        try  
        {  
            URI myURI = URI.create("file:///SDCard/Databases/SQLite_Guide/" +  
                                   "MyTestDatabase.db");
```

```
        d = DatabaseFactory.open(myURI);

        Statement st = d.createStatement("INSERT INTO People(Name,Age) " +
                                       "VALUES ('John',37)");

        st.prepare();
        st.execute();
        st.close();
        d.close();

    }
    catch ( Exception e )
    {
        System.out.println( e.getMessage() );
        e.printStackTrace();
    }
}
}
```

Code sample: Creating a parameterized insert

```
/*
 * ParameterizedInsert.java
 *
 * Research In Motion Limited proprietary and confidential
 * Copyright Research In Motion Limited, 2010
 */
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.database.*;
import net.rim.device.api.io.*;
import java.util.*;

public class ParameterizedInsert extends UiApplication
{
    public static void main(String[] args)
    {
        ParameterizedInsert theApp = new ParameterizedInsert();
        theApp.enterEventDispatcher();
    }
    public ParameterizedInsert()
    {
        pushScreen(new ParameterizedInsertScreen());
    }
}
class ParameterizedInsertScreen extends MainScreen
```

```
{
Database d;
public ParameterizedInsertScreen()
{
    LabelField title = new LabelField("SQLite Insert Data " +
        "Schema Sample",
        LabelField.ELLIPSIS |
        LabelField.USE_ALL_WIDTH);
    setTitle(title);
    add(new RichTextField("Attempting to insert data into " +
        "MyTestDatabase.db on the SDCard."));
    try
    {
        URI myURI = URI.create("file:///SDCard/Databases/SQLite_Guide/" +
            "MyTestDatabase.db");
        d = DatabaseFactory.open(myURI);

        Statement st = d.createStatement("INSERT INTO People(Name,Age) " +
            "VALUES (?,?)");
        st.prepare();

        Hashtable ht = new Hashtable(4);
        ht.put("Sophie", new Integer(6));
        ht.put("Karen", new Integer(3));
        ht.put("Kevin", new Integer(82));
        ht.put("Cindy", new Integer(12));

        Enumeration names = ht.keys();
        Enumeration ages = ht.elements();

        while (names.hasMoreElements())
        {
            String strName = (String)names.nextElement();
            Integer iAge = (Integer)ages.nextElement();
            st.bind(1,strName);
            st.bind(2,iAge.intValue());
            st.execute();
            st.reset();
        }
        st.close();
        d.close();
    }
    catch ( Exception e )
    {
        System.out.println( e.getMessage() );
        e.printStackTrace();
    }
}
}
```

Code sample: Retrieving table data

```
/*
 * ReadData.java
 *
 * Research In Motion Limited proprietary and confidential
 * Copyright Research In Motion Limited, 2010
 */
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.database.*;
import net.rim.device.api.io.*;

public class ReadData extends UiApplication
{
    public static void main(String[] args)
    {
        ReadData theApp = new ReadData();
        theApp.enterEventDispatcher();
    }

    public ReadData()
    {
        pushScreen(new ReadDataScreen());
    }
}

class ReadDataScreen extends MainScreen
{
    Database d;
    public ReadDataScreen()
    {
        LabelField title = new LabelField("SQLite Read Table Data Sample",
                                           LabelField.ELLIPSIS |
                                           LabelField.USE_ALL_WIDTH);

        setTitle(title);
        add(new RichTextField("Attempting to retrieve data from " +
                              "MyTestDatabase.db on the SDCard."));

        try
        {
            URI myURI = URI.create("file:///SDCard/Databases/SQLite_Guide/" +
                                   "MyTestDatabase.db");
            d = DatabaseFactory.open(myURI);

            Statement st = d.createStatement("SELECT Name, Age FROM People");

            st.prepare();
        }
    }
}
```

```

        net.rim.device.api.database.Cursor c = st.getCursor();

        Row r;
        int i = 0;
        while(c.next())
        {
            r = c.getRow();
            i++;
            add(new RichTextField(i + ". Name = " + r.getString(0) +
                ", " +
                "Age = " + r.getInteger(1)));
        }
        if (i==0)
        {
            add(new RichTextField("No data in the People table.));
        }
        st.close();
        d.close();

    }
    catch ( Exception e )
    {
        System.out.println( e.getMessage() );
        e.printStackTrace();
    }
}
}

```

Code sample: Deleting table data

```

/*
 * DeleteData.java
 *
 * Research In Motion Limited proprietary and confidential
 * Copyright Research In Motion Limited, 2010
 */
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.database.*;
import net.rim.device.api.io.*;

public class DeleteData extends UiApplication
{
    public static void main(String[] args)
    {
        DeleteData theApp = new DeleteData();
    }
}

```

```

        theApp.enterEventDispatcher();
    }

    public DeleteData()
    {
        pushScreen(new DeleteDataScreen());
    }
}

class DeleteDataScreen extends MainScreen
{
    Database d;
    public DeleteDataScreen()
    {
        LabelField title = new LabelField("SQLite Delete Database Data",
                                          LabelField.ELLIPSIS |
                                          LabelField.USE_ALL_WIDTH);

        setTitle(title);
        add(new RichTextField("Attempting to delete data from " +
                              "MyTestDatabase.db on the SDCard."));

        try
        {
            URI myURI = URI.create("file:///SDCard/Databases/SQLite_Guide/" +
                                   "MyTestDatabase.db");
            d = DatabaseFactory.open(myURI);

            Statement st = d.createStatement("DELETE People");
            st.prepare();
            st.execute();
            st.close();
            d.close();

        }
        catch ( Exception e )
        {
            System.out.println( e.getMessage() );
            e.printStackTrace();
        }
    }
}
}

```

Code sample: Updating table data

```

/*
 * UpdateData.java
 *
 * Research In Motion Limited proprietary and confidential

```

```
* Copyright Research In Motion Limited, 2010
*/
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.database.*;
import net.rim.device.api.io.*;

public class UpdateData extends UiApplication
{
    public static void main(String[] args)
    {
        UpdateData theApp = new UpdateData();
        theApp.enterEventDispatcher();
    }

    public UpdateData()
    {
        pushScreen(new UpdateDataScreen());
    }
}

class UpdateDataScreen extends MainScreen
{
    Database d;
    public UpdateDataScreen()
    {
        LabelField title = new LabelField("SQLite Update Data Sample",
                                           LabelField.ELLIPSIS |
                                           LabelField.USE_ALL_WIDTH);

        setTitle(title);
        add(new RichTextField("Trying to update data in MyTestDatabase.db.));
        try
        {
            URI myURI = URI.create("file:///SDCard/Databases/SQLite_Guide/" +
                                   "MyTestDatabase.db");
            d = DatabaseFactory.open(myURI);

            Statement st = d.createStatement("UPDATE People SET Age=38 " +
                                             "WHERE Name='John'");

            st.prepare();
            st.execute();
            st.close();
            d.close();

        }
        catch ( Exception e )
        {
            System.out.println( e.getMessage() );
            e.printStackTrace();
        }
    }
}
```

```

    }
}

```

Code sample: Creating a parameterized update

```

/*
 * ParameterizedUpdate.java
 *
 * Research In Motion Limited proprietary and confidential
 * Copyright Research In Motion Limited, 2010
 */
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.database.*;
import net.rim.device.api.io.*;
import java.util.*;

public class ParameterizedUpdate extends UiApplication
{
    public static void main(String[] args)
    {
        ParameterizedUpdate theApp = new ParameterizedUpdate();
        theApp.enterEventDispatcher();
    }
    public ParameterizedUpdate()
    {
        pushScreen(new ParameterizedUpdateScreen());
    }
}
class ParameterizedUpdateScreen extends MainScreen
{
    Database d;
    public ParameterizedUpdateScreen()
    {
        LabelField title = new LabelField("SQLite Parameterized Update Sample",
        LabelField.ELLIPSIS |
        LabelField.USE_ALL_WIDTH);
        setTitle(title);

        add(new RichTextField("Attempting to update data in " +
        "MyTestDatabase.db on the SDCard."));
        try
        {
            URI myURI = URI.create("file:///SDCard/Databases/SQLite_Guide/" +
            "MyTestDatabase.db");
            d = DatabaseFactory.open(myURI);

```

```

Statement st = d.createStatement("UPDATE People SET Age=? WHERE Name=?");
st.prepare();

Hashtable ht = new Hashtable(2);
ht.put("Sophie", new Integer(10));
ht.put("Karen", new Integer(7));

Enumeration names = ht.keys();
Enumeration ages = ht.elements();

while (names.hasMoreElements())
{
    Integer iAge = (Integer)ages.nextElement();
    String strName = (String)names.nextElement();
    st.bind(1, iAge.intValue());
    st.bind(2, strName);
    st.execute();
    st.reset();
}
st.close();
d.close();
}
catch ( Exception e )
{
    System.out.println( e.getMessage() );
    e.printStackTrace();
}
}
}

```

Code sample: Deleting a SQLite database

```

/*
 * DeleteDatabase.java
 *
 * Research In Motion Limited proprietary and confidential
 * Copyright Research In Motion Limited, 2010
 */
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.database.*;
import net.rim.device.api.io.*;

public class DeleteDatabase extends UiApplication
{
    public static void main(String[] args)

```

```

    {
        DeleteDatabase theApp = new DeleteDatabase();
        theApp.enterEventDispatcher();
    }

    public DeleteDatabase()
    {
        pushScreen(new DeleteDatabaseScreen());
    }
}

class DeleteDatabaseScreen extends MainScreen
{
    Database d;
    public DeleteDatabaseScreen()
    {
        LabelField title = new LabelField("SQLite Delete Database Sample",
                                           LabelField.ELLIPSIS |
                                           LabelField.USE_ALL_WIDTH);

        setTitle(title);
        add(new RichTextField("Deleting a database called " +
                              "MyTestDatabase.db on the SDCard.));

        try
        {
            URI myURI = URI.create("file:///SDCard/Databases/SQLite_Guide/" +
                                    "MyTestDatabase.db");
            DatabaseFactory.delete(myURI);
        }
        catch ( Exception e )
        {
            System.out.println( e.getMessage() );
            e.printStackTrace();
        }
    }
}
}

```

Code sample: Listing database tables

```

/*
 * ListTables.java
 *
 * Research In Motion Limited proprietary and confidential
 * Copyright Research In Motion Limited, 2010
 */
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.database.*;

```

```

import net.rim.device.api.io.*;

public class ListTables extends UiApplication
{
    public static void main(String[] args)
    {
        ListTables theApp = new ListTables();
        theApp.enterEventDispatcher();
    }

    public ListTables()
    {
        pushScreen(new ListTablesScreen());
    }
}

class ListTablesScreen extends MainScreen
{
    Database d;
    public ListTablesScreen()
    {
        LabelField title = new LabelField("SQLite List Database Tables",
                                          LabelField.ELLIPSIS |
                                          LabelField.USE_ALL_WIDTH);

        setTitle(title);
        add(new RichTextField("Attempting to list tables in " +
                             "MyTestDatabase.db on the SDCard."));

        try
        {
            URI myURI = URI.create("file:///SDCard/Databases/SQLite_Guide/" +
                                   "MyTestDatabase.db");
            d = DatabaseFactory.open(myURI);
            Statement st = d.createStatement("SELECT name FROM " +
                                           "sqlite_master " +
                                           "WHERE type='table' " +
                                           "ORDER BY name");

            st.prepare();
            net.rim.device.api.database.Cursor c = st.getCursor();

            Row r;
            int i = 0;
            while(c.next())
            {
                r = c.getRow();
                i++;
                add(new RichTextField(i + ". Table: " + r.getString(0)));
            }
            if (i==0)
            {
                add(new RichTextField("There are no tables " +
                                       " in the MyTestDatabase database."));
            }
        }
    }
}

```

```

        }
        st.close();
        d.close();

    }
    catch ( Exception e )
    {
        System.out.println( e.getMessage() );
        e.printStackTrace();
    }
}
}

```

Code sample: Using transactions

```

/*
 * UsingTransactions.java
 *
 * Research In Motion Limited proprietary and confidential
 * Copyright Research In Motion Limited, 2010
 */
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.database.*;
import net.rim.device.api.io.*;
public class UsingTransactions extends UiApplication
{
    public static void main(String[] args)
    {
        UsingTransactions theApp = new UsingTransactions();
        theApp.enterEventDispatcher();
    }
    public UsingTransactions()
    {
    }
}

class UsingTransactionsScreen extends MainScreen
{
    Database d;
    public UsingTransactionsScreen()
    {
        LabelField title = new LabelField("SQLite Using Transactions Sample",
            LabelField.ELLIPSIS |
            LabelField.USE_ALL_WIDTH);
    }
}

```

```
        setTitle(title);
        add(new RichTextField("Trying to update data in a single transaction in
MyTestDatabase.db."));
        try
        {
            URI myURI = URI.create("file:///SDCard/Databases/SQLite_Guide/" +
                "MyTestDatabase.db");
            d = DatabaseFactory.open(myURI);

            d.beginTransaction();
            Statement st = d.createStatement("UPDATE People SET Age=7 " +
                "WHERE Name='Sophie'");
            st.prepare();
            st.execute();
            st.reset();
            st = d.createStatement("UPDATE People SET Age=4 " +
                "WHERE Name='Karen'");
            st.prepare();
            st.execute();
            d.commitTransaction();
            st.close();
            d.close();
        }
        catch ( Exception e )
        {
            System.out.println( e.getMessage() );
            e.printStackTrace();
        }
    }
}
```

Glossary

8

API

application programming interface

SQL

Structured Query Language

Provide feedback

9

To provide feedback on this deliverable, visit www.blackberry.com/docsfeedback.

Document revision history

10

Date	Description
18 August 2009	Added a topic about the VACUUM command.

Legal notice

11

©2010 Research In Motion Limited. All rights reserved. BlackBerry®, RIM®, Research In Motion®, SureType®, SurePress™ and related trademarks, names, and logos are the property of Research In Motion Limited and are registered and/or used in the U.S. and countries around the world.

Java is a trademark of Sun Microsystems, Inc. SQLite is a trademark of Hipp, Wyrick & Company, Inc. All other trademarks are the property of their respective owners.

This documentation including all documentation incorporated by reference herein such as documentation provided or made available at www.blackberry.com/go/docs is provided or made accessible "AS IS" and "AS AVAILABLE" and without condition, endorsement, guarantee, representation, or warranty of any kind by Research In Motion Limited and its affiliated companies ("RIM") and RIM assumes no responsibility for any typographical, technical, or other inaccuracies, errors, or omissions in this documentation. In order to protect RIM proprietary and confidential information and/or trade secrets, this documentation may describe some aspects of RIM technology in generalized terms. RIM reserves the right to periodically change information that is contained in this documentation; however, RIM makes no commitment to provide any such changes, updates, enhancements, or other additions to this documentation to you in a timely manner or at all.

This documentation might contain references to third-party sources of information, hardware or software, products or services including components and content such as content protected by copyright and/or third-party web sites (collectively the "Third Party Products and Services"). RIM does not control, and is not responsible for, any Third Party Products and Services including, without limitation the content, accuracy, copyright compliance, compatibility, performance, trustworthiness, legality, decency, links, or any other aspect of Third Party Products and Services. The inclusion of a reference to Third Party Products and Services in this documentation does not imply endorsement by RIM of the Third Party Products and Services or the third party in any way.

EXCEPT TO THE EXTENT SPECIFICALLY PROHIBITED BY APPLICABLE LAW IN YOUR JURISDICTION, ALL CONDITIONS, ENDORSEMENTS, GUARANTEES, REPRESENTATIONS, OR WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY CONDITIONS, ENDORSEMENTS, GUARANTEES, REPRESENTATIONS OR WARRANTIES OF DURABILITY, FITNESS FOR A PARTICULAR PURPOSE OR USE, MERCHANTABILITY, MERCHANTABLE QUALITY, NON-INFRINGEMENT, SATISFACTORY QUALITY, OR TITLE, OR ARISING FROM A STATUTE OR CUSTOM OR A COURSE OF DEALING OR USAGE OF TRADE, OR RELATED TO THE DOCUMENTATION OR ITS USE, OR PERFORMANCE OR NON-PERFORMANCE OF ANY SOFTWARE, HARDWARE, SERVICE, OR ANY THIRD PARTY PRODUCTS AND SERVICES REFERENCED HEREIN, ARE HEREBY EXCLUDED. YOU MAY ALSO HAVE OTHER RIGHTS THAT VARY BY STATE OR PROVINCE. SOME JURISDICTIONS MAY NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES AND CONDITIONS. TO THE EXTENT PERMITTED BY LAW, ANY IMPLIED WARRANTIES OR CONDITIONS RELATING TO THE DOCUMENTATION TO THE EXTENT THEY CANNOT BE EXCLUDED AS SET OUT ABOVE, BUT CAN BE LIMITED, ARE HEREBY LIMITED TO NINETY (90) DAYS FROM THE DATE YOU FIRST ACQUIRED THE DOCUMENTATION OR THE ITEM THAT IS THE SUBJECT OF THE CLAIM.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN YOUR JURISDICTION, IN NO EVENT SHALL RIM BE LIABLE FOR ANY TYPE OF DAMAGES RELATED TO THIS DOCUMENTATION OR ITS USE, OR PERFORMANCE OR NON-PERFORMANCE OF ANY SOFTWARE, HARDWARE, SERVICE, OR ANY THIRD PARTY PRODUCTS AND SERVICES REFERENCED HEREIN INCLUDING WITHOUT LIMITATION ANY OF THE FOLLOWING DAMAGES: DIRECT, CONSEQUENTIAL, EXEMPLARY, INCIDENTAL, INDIRECT, SPECIAL, PUNITIVE, OR AGGRAVATED DAMAGES, DAMAGES FOR LOSS OF PROFITS OR REVENUES, FAILURE TO REALIZE ANY EXPECTED SAVINGS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, LOSS OF

BUSINESS OPPORTUNITY, OR CORRUPTION OR LOSS OF DATA, FAILURES TO TRANSMIT OR RECEIVE ANY DATA, PROBLEMS ASSOCIATED WITH ANY APPLICATIONS USED IN CONJUNCTION WITH RIM PRODUCTS OR SERVICES, DOWNTIME COSTS, LOSS OF THE USE OF RIM PRODUCTS OR SERVICES OR ANY PORTION THEREOF OR OF ANY AIRTIME SERVICES, COST OF SUBSTITUTE GOODS, COSTS OF COVER, FACILITIES OR SERVICES, COST OF CAPITAL, OR OTHER SIMILAR PECUNIARY LOSSES, WHETHER OR NOT SUCH DAMAGES WERE FORESEEN OR UNFORESEEN, AND EVEN IF RIM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN YOUR JURISDICTION, RIM SHALL HAVE NO OTHER OBLIGATION, DUTY, OR LIABILITY WHATSOEVER IN CONTRACT, TORT, OR OTHERWISE TO YOU INCLUDING ANY LIABILITY FOR NEGLIGENCE OR STRICT LIABILITY.

THE LIMITATIONS, EXCLUSIONS, AND DISCLAIMERS HEREIN SHALL APPLY: (A) IRRESPECTIVE OF THE NATURE OF THE CAUSE OF ACTION, DEMAND, OR ACTION BY YOU INCLUDING BUT NOT LIMITED TO BREACH OF CONTRACT, NEGLIGENCE, TORT, STRICT LIABILITY OR ANY OTHER LEGAL THEORY AND SHALL SURVIVE A FUNDAMENTAL BREACH OR BREACHES OR THE FAILURE OF THE ESSENTIAL PURPOSE OF THIS AGREEMENT OR OF ANY REMEDY CONTAINED HEREIN; AND (B) TO RIM AND ITS AFFILIATED COMPANIES, THEIR SUCCESSORS, ASSIGNS, AGENTS, SUPPLIERS (INCLUDING AIRTIME SERVICE PROVIDERS), AUTHORIZED RIM DISTRIBUTORS (ALSO INCLUDING AIRTIME SERVICE PROVIDERS) AND THEIR RESPECTIVE DIRECTORS, EMPLOYEES, AND INDEPENDENT CONTRACTORS.

IN ADDITION TO THE LIMITATIONS AND EXCLUSIONS SET OUT ABOVE, IN NO EVENT SHALL ANY DIRECTOR, EMPLOYEE, AGENT, DISTRIBUTOR, SUPPLIER, INDEPENDENT CONTRACTOR OF RIM OR ANY AFFILIATES OF RIM HAVE ANY LIABILITY ARISING FROM OR RELATED TO THE DOCUMENTATION.

Prior to subscribing for, installing, or using any Third Party Products and Services, it is your responsibility to ensure that your airtime service provider has agreed to support all of their features. Some airtime service providers might not offer Internet browsing functionality with a subscription to the BlackBerry® Internet Service. Check with your service provider for availability, roaming arrangements, service plans and features. Installation or use of Third Party Products and Services with RIM's products and services may require one or more patent, trademark, copyright, or other licenses in order to avoid infringement or violation of third party rights. You are solely responsible for determining whether to use Third Party Products and Services and if any third party licenses are required to do so. If required you are responsible for acquiring them. You should not install or use Third Party Products and Services until all necessary licenses have been acquired. Any Third Party Products and Services that are provided with RIM's products and services are provided as a convenience to you and are provided "AS IS" with no express or implied conditions, endorsements, guarantees, representations, or warranties of any kind by RIM and RIM assumes no liability whatsoever, in relation thereto. Your use of Third Party Products and Services shall be governed by and subject to you agreeing to the terms of separate licenses and other agreements applicable thereto with third parties, except to the extent expressly covered by a license or other agreement with RIM.

Certain features outlined in this documentation require a minimum version of BlackBerry® Enterprise Server, BlackBerry® Desktop Software, and/or BlackBerry® Device Software.

The terms of use of any RIM product or service are set out in a separate license or other agreement with RIM applicable thereto. NOTHING IN THIS DOCUMENTATION IS INTENDED TO SUPERSEDE ANY EXPRESS WRITTEN AGREEMENTS OR WARRANTIES PROVIDED BY RIM FOR PORTIONS OF ANY RIM PRODUCT OR SERVICE OTHER THAN THIS DOCUMENTATION.

Research In Motion Limited
295 Phillip Street

Waterloo, ON N2L 3W8
Canada

Research In Motion UK Limited
Centrum House
36 Station Road
Egham, Surrey TW20 9LF
United Kingdom

Published in Canada