Archived:Using Symbian SQL API with scalar queries

Archived: This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

Overview

The `TSqlScalarFullSelectQuery` class is used in situations where the SELECT SQL query refering to several data fields returns only one row with a single column value. When compared to class `RSqlStatement`, `TSqlScalarFullSelectQuery` is more efficient and recommended to use when possible. The class instance is created with a parameter that refers to the opened database. `TSqlFullSelectQuery` has a number of methods that take the SELECT SQL query as a parameter and return values of different data types. This snippet shows how to use methods `TSqlScalarFullSelectQuery::SelectIntL()` and `TSqlScalarFullSelectQuery::SelectTextL()`.

This snippet can be self-signed.

MMP file

The following libraries are required:

```
LIBRARY euser.lib
LIBRARY sqldb.lib
```

The following capabilities are needed to test the example secure database:

```
CAPABILITY ReadUserData
CAPABILITY WriteUserData
```

Preconditions

Databases nonsecure.db and [UID3]secure.db need to be created before executing this code snippet. See Archived:Using SQL API for creating non-secure and secure databases on Symbian.

Source file

```
#include <e32base.h>
#include <SqlDb.h>

void ScalarQueriesL()
  {
  RSqlDatabase database;
  _LIT(KNonSecureDbName, "\\nonsecure.db");
  _LIT(KSecureDbName, "[E80000AF]secure.db");

  TInt error = database.Open(KNonSecureDbName);

  if (error == KErrNone)
    {
    CleanupClosePushL(database);

    //count how many rows is in the database
    _LIT(KSqlSelectCount,"SELECT COUNT (*) FROM MOVIES");
    TSqlScalarFullSelectQuery sqlSelectCountQuery(database);
    TInt count = sqlSelectCountQuery.SelectIntL(KSqlSelectCount);

    //get the max year value (the newest movies)
    _LIT(KSqlSelectMax,"SELECT MAX(YEAR) FROM MOVIES");
    TSqlScalarFullSelectQuery sqlSelectMaxQuery(database);
    TInt max = sqlSelectMaxQuery.SelectIntL(KSqlSelectMax);

    //get the average year value
    _LIT(KSqlSelectAvg,"SELECT AVG(YEAR) FROM MOVIES");
    TSqlScalarFullSelectQuery sqlSelectAvgQuery(database);
    TInt avg = sqlSelectAvgQuery.SelectIntL(KSqlSelectAvg);

    CleanupStack::PopAndDestroy(); //database
    }
  else
    {
    //open database failed
    }

  error = database.Open(KSecureDbName);

  if (error == KErrNone)
```


(C) Copyright Nokia 2013. All rights reserved.

```
    {
    CleanupClosePushL(database);

    _LIT(KSqlSelectFirstBook,"SELECT TITLE FROM BOOKS WHERE ID=1");

    TSqlScalarFullSelectQuery fullSelectQuery(database);
    HBufC* titleBuf = HBufC::NewLC(5);
    TPtr title = titleBuf->Des();
    TInt ret = fullSelectQuery.SelectTextL(KSqlSelectFirstBook, title);

    //if ret > 0 buffer reallocation is needed
    if(ret > 0)
      {
      titleBuf = titleBuf->ReAllocL(ret);
      CleanupStack::Pop();
      CleanupStack::PushL(titleBuf);
      title.Set(titleBuf->Des());
      ret = fullSelectQuery.SelectTextL(KSqlSelectFirstBook, title);
      //do something with the data...
      }
    else
      {
      //do something with the data...
      }

    CleanupStack::PopAndDestroy(2);//buf, database
    }
  else
    {
    //open database failed
    }


  }
```

Postconditions

The `TSqlScalarFullSelectQuery` class is used to get COUNT, MAX, and AVG values from the database nonsecure.db and one specific textual value from the dateabase [UID3]secure.db.

See also

- Archived:Using SQL API for creating non-secure and secure databases on Symbian
- Archived:Using Symbian SQL API for attaching and detaching databases
- Archived:Using Symbian SQL API with SQL statements which do not return data
- Archived:Using Symbian SQL API with SQL statements which return data
- Archived:Using Symbian SQL API with data streams

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.