

DBMS

Reviewer Approved

Introduction

Symbian OS DBMS provides features for creating and maintaining databases, and implements reliable and secure data access to these databases via both native and SQL calls. These calls are supported by a transaction/rollback mechanism that ensures that either all data is written or none at all.

DBMS on the Symbian OS is a powerful, lightweight and supports normal add/search/retrieve/ update/delete functionality as well as basic Structured Query Language (SQL), Data Definition Language (DDL), and Data Modeling Language (DML) statement handling.

Create a Database

Create a database using RDbStoreDatabase The following code snippet shows the creation of a database using RDbStoreDatabase. It creates a File Store object and constructs a database within it.

Headers Required:

```
#include <f32file.h> //RFs
#include <d32dbms.h> //RDbStoreDatabase, RDbNamedDatabase
#include <s32file.h> //CFileStore
```

Library required:

```
LIBRARY efsrv.lib //RFs
LIBRARY edbms.lib //RDbStoreDatabase, RDbNamedDatabase
LIBRARY estor.lib //CFileStore
```

Source:

```
class CBookstoreDb : public CBase
{
    ...
private: // Member data
    RFs iFsSession;
    RDbStoreDatabase iBookstoreDb;
    CFileStore* iFileStore;
    ...
};
TInt CBookstoreDb::CreateDbL(const TFileName& aNewBookstoreFile)
{
    User::LeaveIfError(iFsSession.Connect());
    // Create new or replace existing file
    iFileStore = CPermanentFileStore::ReplaceL(iFsSession,
        aNewBookstoreFile, EFileRead|EFileWrite);
    iFileStore->SetTypeL(iFileStore->Layout());
    TStreamId id = iBookstoreDb.CreateL(iFileStore);
    iFileStore->SetRootL(id); // Keep database id as root of store
    iFileStore->CommitL(); // Complete creation by committing
    ... // Database is now open and ready for operations.
```

NOTE: CreateDbL creates a new database file. It takes TFileName, which is the full filename (including path). The example overwrites any previous file of the same name (ReplaceL()). Finally, it creates the root stream object and commits the database structure.

Create a database using RDbNamedDatabase The following code shows database creation using RDbNamedDatabase, which is simpler, because there is no need to work with streams:

```
...
Rfs iFsSession;
RDbNamedDatabase iBookstoreDb;
...
TInt CBookstoreDb::CreateDbL(const TFileName& aNewBookstoreFile)
{
    User::LeaveIfError(iFsSession.Connect());
    // Create new or replace existing database
    User::LeaveIfError(iBookstoreDb.Replace(fsSession,
    aNewBookstoreFile));
    ... // Database is now open. Create tables etc.
```

The Replace method creates a new database file or replaces an existing one. It also formats the file as an empty database. The database is then open and ready for creating table structures, etc. Note that by creating a database, the access mode is exclusive. Replace takes in a TFileName, which is a full filename (including path). In Symbian OS DBMS, DBMS names are limited in length to 64 characters. The database names correspond to filenames. It is up to the programmer whether or not to use a file extension. As an example, the following names are valid

```
_LIT(KDbName, "C:\\system\\apps\\bookstoredb\\bookstore.dat");
_LIT(KDbName, "C:\\system\\apps\\bookstoredb\\bookstore");
```

Define Tables within a Database

To define tables in a database, the developer needs to be aware of the three key API concepts: column, column set, and index key.

The index key is encapsulated in `CDbKey`. A column for the key is encapsulated in `TDbKeyCol`. A column definition is encapsulated in `TDbCol`.

Query Schema Information

There are APIs to query database structure (indexes, tables, and their structures). The database base class, `RdbDatabase`, provides the base means:

- `TableNamesL()`: List of table names encapsulated in `CDbTableNames`.
- `IndexNamesL()`: List of index names encapsulated in `CDbIndexNames`. The method takes in a table name.
- `ColSetL()`: Column definitions for a table encapsulated in `CDbColSet`. The method takes in a table name.
- `KeyL()`: Index definition for an index within a table. The method takes in a table name and an index name. The column definitions `CDbColSet` for a table can be also queried from a table or an SQL view using their base class method `RdbRowSet::ColSetL()`. The `CDbColSet` can be iterated using its methods or utilizing `TDbColSetIter` class.

Open and Close a Database

The database can be opened in an exclusive client-side mode and in a shared client/server mode. `RdbNamedDatabase` supports both modes, but the `RdbStoreDatabase` API provides the means to open a database in the client-side mode only.

It is recommended to use the `RdbNamedDatabase` API. The client-side access mode is slightly more efficient, but the database is not accessible to other applications because the file is locked. In a typical application, the client/server mode is recommended.

Create Data

Add a new row to database involves first inserting an empty row, updating values for the row, and finally completing insertion to the database. `table.InsertL()` inserts an empty row and `table.SetCol(...)` sets the data. If the description column is long, it requires the use of `RdbColWriteStream`.

Related Links

- [Database Example](#)
- [Database Example with Navigation](#)
- [Demonstrates basic use of the Symbian OS DBMS](#) 
- [S60 Platform: Using DBMS APIs v2.0 \(PDF\)](#) 
- [Regan Coleman. Symbian Database Components. A small footprint DBMS for small footprint devices.](#) 