TDDBUtils

Version 1.5.2
developed by © TechDoc G. Becker

for

SQLite 3 or SQLCipher 4 Databases

http://www.gbecker.de

This library is under Construction! Please report any errors as Post via B4X Forum or direct to techdoc@gbecker.de.

Inhalt

1 Copyright, Warranty, Support

© Free for personnel use.
© For commercial use please contact us on techdoc@gbecker.de

Notice! Usage of the Code is under your own risk!

This library and code is not supported for free.

2 Overview

This is a B4A Code Module / B4X-Library. It supports the developer by the handling of data bound views.

Function overview:

OpenDB	Copies DB To accessible Directory and opens it
CloseDB	Closes the DB
startTransaction	start a batch operation
commitTransaction	Do all actions And close batch operation
setDBcompactMode	set the Compact Mode
compactDB	Do shrinking
Table Names	get the table names
ColumnsInfo	get Info of all columns
findColumnInfo	get info of a named column
insert	insert new record
update	update a record
delete	delete a record
save	insert record or update record If it exists
select1	simple Select
select2	Select over joined tables
select3	Select from/til a date
select4	report Max/Min/SUm/count value of a select
dateDifference	Select between To dates
checkDBNull	check If a column value is Null
checkExists	check If a record exists
EncryptText	encrypt a string
DecryptText	decrypt a string
clearViews	clearViews of a Form
Views2Columns	Transfer Views values of a form to data Columns
ColumnstViews	transfer data columns value to form views values

3 Notice

This TDDBUtils functions deal only with one single Database.

The library declares a public global variable data type TYPE.

```
Type TColInfo (CID As Int,Name As String, _
               Typ As String, NotNull As Boolean, PK As Boolean)
```

The library uses this additional libraries: XUI, StringUtils, SQL, SQLCipher.

Install (copy) the library TDDBUtils.b4xlib in your additional libraries folder.

Every function assumes that the database is successful opened (function *openDB*) before it is used.

Table or Column Names that include a space character must be enclosed in brackets like [Article Number]. The names are case sensitive.

To build the *Condition* Parameter (SQL WHERE Clause). Do not start with the code word WHERE. If you are searching for characters (Text) you must include the characters with high comma like Name='Joe'. It is allowed to use Wildcards (%,?).

Condition Examples:
ArticleNo=5
ArticleDesc='Apple'
FileName LIKE '%.png'
Name LIKE 'B?ker%'

You are also allowed to use all operands like (<,>,<>,=,>=,<=) and to build a chain with (AND, OR, NOT).

All *Date*, *Time* and *DateTime* data is stored in the database as a special numerical *Tick* value type *INTEGER*. The affiliate Data Types *Date*, Time and *DateTime* are internally converted to the numerical value by the database engine itself if they are given in a correct string format. For Date and Time string Formats see Documentation of *DateTime.DateFormat* and *DateTime.TimeFormat.*
The standard formats of the Database are YYYY-MM-dd HH:mm:ss. Where YYYY = 4 digit year, MM = 2 digit Month, dd = 2 digit Day, HH = 24 Hour 2 digit Hour, mm = 2 digit minute, ss = 2 digit second.
To operate with the numerical values, use the internal *DateTime* Functions.

Boolean values are stored as 1 (*True*) or 0 (*False*) numerical value in a data column type *INTEGER*, the affiliate is *BOOL*.

Images or encrypted Text is stored as Bytes in a data column type *BLOB*.

To manage the Database on PC Desktop we recommend to use the free Application *DB* Browser. To get an encrypted *SQLite* Database you must first create a normal *SQLite* Database. Open this Database in *DB Browser* and use the *DB Browers* Function to convert it into an encrypted Database. B4X SQLCipher library is an enhancement to *B4X SQL library*.

See Appendix for a small tutorial.

4 Database

4.1 OpenDB

OpenDB(DBName As String, Password As String,DeleteDB As Boolean) As Boolean

Copy the database file from *DirAssets* to an accessible directory *DirInternal*. If ordered delete an existing database at this place.

DBName	the filename like abc.db
Password	**the password to be used for SQLCipher-DB. If you like to use SQLite-DB leave parameter empty.**
DeleteDB	Delete an existing database file at the accessible directory. This must be used with value True if the database file in the asset directory has been modified. **Notice!** Normally this value must be False to avoid loss of data. Remember the database file in the asset directory is only the template for the copying to the accessible directory. The database your App is working with is stored in *DirInternal*.

If you like to work with *SQLite* leave the password empty. If the password has a value working with *SQLCipher* is assumed.

If the file was found and copied and the database access is ok (database is open) than *True* is returned, otherwise *False*.

4.2 CloseDB

CloseDB As Boolean

Close an open Database correctly.

If database is successfully opend *True* is returned otherwise *False*.

Notice! Do not close the App before closing an open database. Loss of data may occur.

4.3 startTransaction

startTransaction As Boolean

Start/Begin a batch operation of database insert/update/delete of records. This is often used if you make a mass of data modifications to make the process quicker. All these operations are filled in stack.

If Transaction is successfully started *True* is returned otherwise *False*.

Notice! To really do the batched actions you have to call '*commitTransaction*'.

4.4 commitTransaction

commitTransaction As Boolean

End Batch operation and release all stored actions to be done.

If commit is successful *True* is returned otherwise *False*.

4.5 setDBcompactMode

setDBcompactMode(Action As String) As Boolean

If you delete records the used dataspace is not released. To compact the database (eliminate this space) you may use the function *compactDB*. Before using you have to set one of the possible compact modes with this function.

Action	The mode	
	none	No compacting
	inc	incremental compacting
	full	total compacting (recommended)

If the mode is successfully set *True* is returned otherwise *False*.

4.6 compactDB

compactDB As Boolean

After setting the compact mode with function *setCompactMode* you start compacting with this function. If it's done well *True* is returned otherwise *False*.

4.7 Table Names

getTable Names As List

Retrieve all table names from the database. If successful a *List* with the names is returned otherwise Null is returned

4.8 ColumnsInfo

getColumnsInfo(TabName As String) As Map

Retrieve all columns information of a named database table.

List of information:

CID	columnID
Name	column Name
Typ	column Type
NotNull	condition column should not be Null
PK	condition column is a unique primary key

TabName	Table Name

The information of the columns is stored in a returned map. Key=Column Name, Value = variable type Type.

```
Type TColInfo (CID As Int,Name As String, _
              Typ As String, NotNull As Boolean, PK As Boolean)
```

If there is an error Null is returned

4.9 findColumnInfo

findColumnInfo(Table Name As String,ColName As String) As TColInfo

This is to get the column information of exact one named column.

Table Name	Table name
ColName	Column name

List of information:

CID	columnID
Name	column Name
Typ	column Type
NotNull	condition column should not be Null
PK	condition column is a unique primary key

The column information is returned with the variable type

```
Type TColInfo (CID As Int,Name As String, _
              Typ As String, NotNull As Boolean, PK As Boolean)
```

If there is an error Null is returned

5 Data management

5.1 Insert

insert(TabName As String,Columns As Map) As Boolean

Insert a new record in the database table.

Table Name	Table Name
Columns	Key=Column Name, Value=Column Value Put only the columns in the map that may have a value.

It is recommended that this is really a new record never been inserted in the database table. If you like to test this before using this function use function *checkExists* before.

Notice! There is not proof of unique columns!

If the record is added *True* is returned otherwise *False*.

5.2 update

update(TabName As String, columns As Map, condition As String) As Boolean

Update the values of an existing (!) record.

TabName	Table Name
Columns	Key=Column Name, Value=Column Value Put only the columns in the map that may have a value.
Condition	The SQL WHERE condition without the word WHERE.

It is recommended that the record should really exists, there is no proof for it. If you like to test this before using this function use function *checkExists* before.

If the record is updated *True* is returned otherwise *False*.

5.3 delete

delete(TabName As String,Condition As String) As Boolean

Delete the selected record. **Deleted records can never been recovered!** There is not security dialog before deleting!

TabName	Table Name
Condition	The SQL WHERE condition without the word WHERE.

It is recommended that the record should really exists, there is no proof for it. If you like to test this before using this function use function *checkExists* before.

if the record is deleted *True* is returned otherwise *False*.

5.4 save

save(TabName As String, columns As Map, Condition As String) As Boolean

This is a combination if the *insert* and the *update* function. At first the function proves whether the record exists or not. If it does an update is done if not an insert.

TabName	Table Name
Columns	Key=Column Name, Value=Column Value Put only the columns in the map that may have a value.
Condition	The SQL WHERE condition without the word WHERE.

If the record is inserted or updated True is returned otherwise False.

6 Data selection

6.1 select1

select1(TabName As String, Columns As String, Condition As String) As ResultSet

This is a simple Data selection for 1 Table. As result a ResultSet is returned otherwise Null is returned.

TabName	Table Name
Columns	Comma separated Column Names. Leave parameter empty to address all columns.
Condition	The SQL WHERE condition without the word WHERE.

6.2 select2

select2(TabName1 As String, ColName1 As String, _
TABName2 As String, ColName2 As String, Condition As String) As ResultSet

Select Data spread over 2 joined tables. As result a ResultSet is returned otherwise Null is returned.

TabName1	Table Name 1st Table
ColName1	Column Name (table connector)
TabName2	Table Name 2nd Table
ColName2	Column Name (table connector)
Condition	The SQL WHERE condition without the word WHERE.

The ColName1 and ColName2 are the columns to connect the to (innerjoin) tables together. Remember the columns must have the same column type and data.

For example:

Table Name1:	ProdDescription
ColName1:	ProdNo
Table Name2:	ProdValues
ColName2:	ProdNo

Call: TDDBUtils.select2(“ProdDescription”,”ProdNo”,*ProdValues”,”ProdNo”,”ProdNo=6000”)

The returned ResultSet has 1 record with the columns of the 2 tables.

6.3 select3

```
select3(TabName As String, ColName As String, _
        startDate As String, _
        Operand As String, _
        nDays As Int,nMonth As Int, nyears As Int) As ResultSet
```

Select records depending on a given date. As result a ResultSet is returned otherwise Null is returned.

TabName	Table Name
ColName	Column Name
startDate	Start date in correct string format corresponding to the DateTime.DateFormat.
Operand	< > <> >= <=
nDays	+/- Days from startDate
nMonth	+/- month from startDate
nYears	+/- Years from startDate

6.4 select4

```
public Sub select4(Action As String,TabName As String, _
        Column Name As String, Condition As String ) As Double
```

Return a calculated value of a named column. If error return Null.

Action	The action name like: Max report highest value Min report minimum value Sum report a sum of all values Avg report the average value Count report the total count of records in the selection
TabName	Table Name
ColName	Column Name
Condition	The SQL WHERE condition without the word WHERE.

6.5 dateDifference

dateDifference(TabName As String, ColName As String, _
startDate As String, endDate As String, _
Condition As String) As ResultSet

Select records between 2 dates. As result a ResultSet is returned otherwise Null is returned.

TabName	Table Name
ColName	Column Name
startDate	Start date in correct string format corresponding to the *DateTime*.*DateFormat*.
endDate	Start date in correct string format corresponding to the *DateTime*.*DateFormat*.
Condition	The SQL WHERE condition without the word WHERE.

6.6 checkDBNull

checkDBNull(rsIn As ResultSet, Pos As Long, _
Column Name As String,ColumnType As String) As Object

Check if the named data column has a Null value. If it is true and the column type is a string an empty value (“”) is returned. If it is type of a number a 0 (zero) value is returned. If it is a blob type a Null Value is returned.

If the column value is not Null the value is returned.

rsIn	The resultset
Pos	Position of the record to proof in the resultset
ColName	Column Name
ColumnType	text int double blob long

6.7 checkExists

checkExists(TabName As String,Condition As String) As Int

Check if a record already exists in a database table.

TabName	Table Name
Condition	The SQL WHERE condition without the word WHERE.

Notice! If you are searching for characters (Text) you must include the characters with high comma like Name=’Joe’. It is allowed to use Wildcards like Name=’%ker’, Label=’%.%’.

If the record exists than the number or records is returned. This may be 1 or >1. If it not exists the returned value is 0. If there was an error -1 is returned.

7 Encrypt/Decrypt Data

7.1 EncryptText

EncryptText(text As String, password As String) As Byte()

Encrypt a given text protected by password and AES encryption.

Text	The text to encrypt
Password	The password for decryption

If encryption is successful a byte array is returned that may be stored in a database column type *Blob.*

7.2 DecryptText

DecryptText(encryptedData() As Byte, password As String) As String

Decrpyt the text encrypted with function *EncryptText*.

Data	A Byte array with the encrypted text may be retrieved from a column type Blob.
Password	The password for decryption

The readable text is returned.

8 Form Generator

The Form Generator

- Clear data bound view values,
- transfers views values to be stored in the database,
- transfers data column values to the view values.

To use the Generator all views of the Form must be a child of Panel. At present this types of view are supported:

- EditText, Label, RadioButton, CheckBox, ImageView.

This data column types are supported:

- TEXT (String), INTEGER, REAL (Double), BLOB, DATE, TIME. DATETIME, BOOL.

The functions next may be modified or enhanced to support more data types or view types.

To recognized child views the Tag property of the child views must contain the correspondent data column name. If you like to have other information stored in the Tag property, please store the data column name at the first position and separate all other information by | (Alt-124). You may use the *regex.split* function to split off the Tag information. All views with an empty Tag value are ignored.

8.1 clearViews

clearViews(Panel As Panel) As Boolean

Clear all views values.

Panel	The panel object holding the child views

If cleared *True* is returned otherwise *False*.

8.2 Views2Columns

Views2Columns(Panel As Panel,TabName As String) As Map

Transfer the views values to the correspondent data columns.

Panel	The panel object holding the child views
TabName	Table Name

A Map is returned. Key=Column Name, Value=Column value.

To insert or update the record in the database use the Map with the functions *insert/update/save*.

8.3 Columns2Views

Columns2Views(Panel As Panel, TabName As String, POS as Long, Recordset As ResultSet) As Boolean

Put the values of the data columns into the child views.

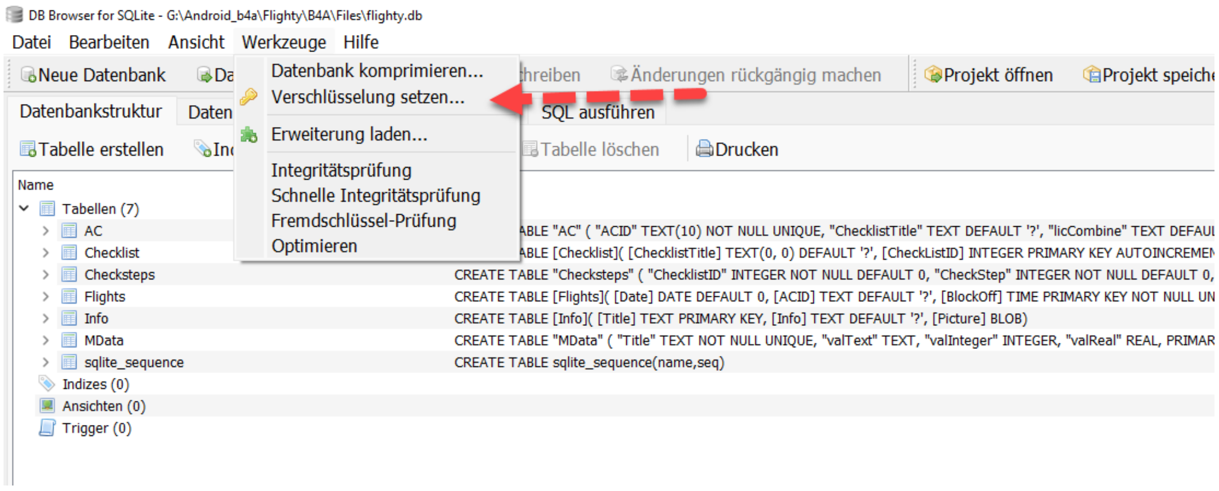
Panel	The panel object holding the child views
TabName	Table Name
Pos	The position of the record in the result set
Recordset	Result set

If the values are transferred *True* is returned otherwise *False*.

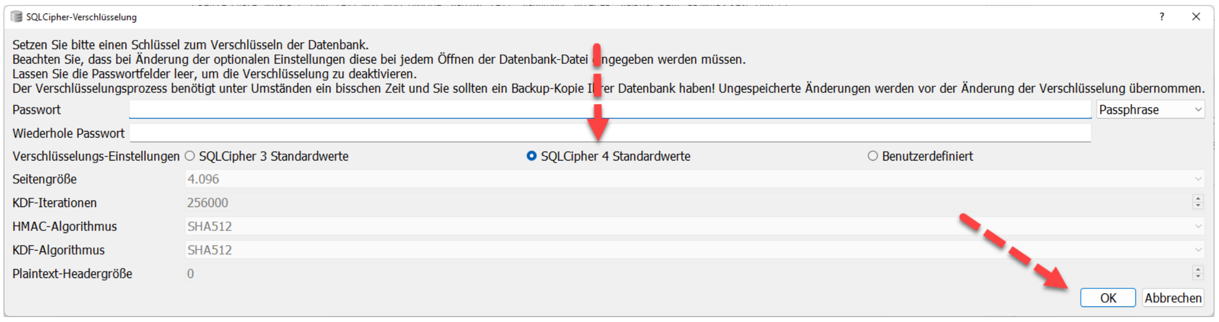
9 Appendix

9.1 SQLite 2 SQLCipher

- Download and install *DBBrowser* (for SQLCipher, encrypted Database) on your PC.
- Open DBBrowser and create a standard SQLite 3 Database.
- Convert Database to be encrypted.



Name	
Tabellen (7)	
AC	ABLE "AC" ("ACID" TEXT(10) NOT NULL UNIQUE, "ChecklistTitle" TEXT DEFAULT '?', "licCombine" TEXT DEFAUL
Checklist	ABLE [Checklist]([ChecklistTitle] TEXT(0, 0) DEFAULT '?', [CheckListID] INTEGER PRIMARY KEY AUTOINCREMEN
Checksteps	CREATE TABLE "Checksteps" ("ChecklistID" INTEGER NOT NULL DEFAULT 0, "CheckStep" INTEGER NOT NULL DEFAULT 0,
Flights	CREATE TABLE [Flights]([Date] DATE DEFAULT 0, [ACID] TEXT DEFAULT '?', [BlockOff] TIME PRIMARY KEY NOT NULL UN
Info	CREATE TABLE [Info]([Title] TEXT PRIMARY KEY, [Info] TEXT DEFAULT '?', [Picture] BLOB)
MData	CREATE TABLE "MData" ("Title" TEXT NOT NULL UNIQUE, "valText" TEXT, "valInteger" INTEGER, "valReal" REAL, PRIMAR
sqlite_sequence	CREATE TABLE sqlite_sequence(name,seq)
Indizes (0)	
Ansichten (0)	
Trigger (0)	



Setzen Sie bitte einen Schlüssel zum Verschlüsseln der Datenbank.
Beachten Sie, dass bei Änderung der optionalen Einstellungen diese bei jedem Öffnen der Datenbank-Datei angegeben werden müssen.
Lassen Sie die Passwortfelder leer, um die Verschlüsselung zu deaktivieren.
Der Verschlüsselungsprozess benötigt unter Umständen ein bisschen Zeit und Sie sollten ein Backup-Kopie Ihrer Datenbank haben! Ungespeicherte Änderungen werden vor der Änderung der Verschlüsselung übernommen.

Passwort	Passphrase
Wiederhole Passwort	
Verschlüsselungs-Einstellungen	SQLCipher 3 Standardwerte [✓ SQLCipher 4 Standardwerte] Benutzerdefiniert
Seitengröße	4.096
KDF-Iterationen	256000
HMAC-Algorithmus	SHA512
KDF-Algorithmus	SHA512
Plaintext-Headergröße	0

- Press OK to encrypt the Database.

If you open an encrypted database you will be ask for the password. Encryption/Decryption will be done one the fly. Notice! You are not able to open an encrypted Database with 'normal' SQLite Tools.

- In B4X add the libraries SQL and SQLCipher to the project.

Notice!

If you lose the password there is no way to open the database and not tool to retrieve the password!