SqlitePass Database components - Major Changes

10/09/2010 – Version 0.55

- Many changes again with this new version and a lot of bugfixes.
- New error dialog with better error handling to let you log and browse the errors history (not finished yet...),
- TSqlitePassDataset has Params support and a new designtime params editor. Params give you access to the queries or sql statements parameters. They are very similar to classic TParams. You can easily retrieve and automatically bind your params to the appropriate field datatype if you suffixe your params names (in the sql statement) like this : MyParamName + As + Datatype.
-

  For example,

  ```
  "select * from MyQuerie where MyFieldDate = :MyParam1AsDate
  and MyFieldCode >= :MyParam2AsInteger;"
  ```

  Supported datatype are AsWidestring, AsString, AsSmallint, AsLargeint, AsInteger, AsWord, AsFloat, AsBoolean, AsCurrency, AsBcd, AsDatetime, AsDate, AsTime, AsVarbytes, AsBytes, AsAutoinc, AsFmtmemo, AsMemo, AsBlob, AsGraphic, AsTypedbinary, AsFixedchar.

  Note : Params values are always filled in as litteral text. Date, DateTime, Time...etc values must follow the corresponding Database.DatatypeOptions.DateFormat, DateTimeFormat, TimeFormat template.

  Here is a small example :

  ```
  DatasetDetail.Close;
  DatasetDetail.SQL.Text := 'SELECT * FROM [DateTime] where Date =:ParamDateAsDate;';
  DatasetDetail.Params.ParamByName('ParamDateAsDate').Value := '2010/12/10';
  DatasetDetail.Open;
  ```

  You can also define a master/detail relation with parameters. At design time, once your datasetname is selected, set the MasterSource Property and then set the params property to show up the parameters dialog. The Value dropdown list allows you to choose the master field for each param you want to link with.

- Better Unicode ↔ Ansi conversions
- TSqlitePassDataset should handle more complexes queries (no thoroughly tested yet)
- and many more...

16/05/2010 – Version 0.51

- The database.options were updated to handle more pragma settings, with a new Property : ApplyMode to control how pragma applies. (see help file).
- Note : I had to change some const TSqlitePassAutoVacuumType   = (avNone, avFull, avIncremental)...etc and you could have some problems reopening existing dfm or lfm files. In this case, open the dfm/lfm as text and remove all the lines related to the database.options, for example Options.AutoVacuum = ….
- **Bugfix** in Filter Text that raised error when dataset was retrevied from *,dfm and already opened.
- **Bugfix :** when a fieldname is quoted with singlequote, it is parsed as plain text string, not indentifier. This is the normal behavior for sqlite.
  I had to remove qsSingleQuote from TsqlitePassQuoteStyle.
  TsqlitePassQuoteStyle is now (qsDoubleQuote, qsBracket, qsGraveQuote, qsNone); This fix is also related to the textfield size errors that could raise in 0.50 version.

- Default library file changed to sqlitepass3.dll to sqlite3.dll or libsqlite3.so
- Log error dialog can be inspected directly from IDE now. (Right click on TsqlitePassDatabase Component). It could help catch bug → Sometimes, database doesn't open, but without raising exception.
- Help file partially updated.
- Some other bugfixes and minor improvements applied to :
    - TSqlitePassFilters and TSqlitePassActiveFilters
    - CreateDatabase dialog
    - TSqlitePassCustomFieldDefs and associated dialog
    - FFieldTypesTranslationRules and associated dialog
    - Null fields
    - Date, Time, DateTime, TimeStamp fields should return the correct data with kexi, Sqlite Administrator, Sqlite Expert, SqliteToolbox databases
    - Record delete operation occuring on an inserted record with auto primary key is allowed now...etc.

10/03/2010 – Version 0.50

This version was focused on data sorting optimizations, the introduction of InMemory indexes, fast LookupFields, reworks on field filters, locate and lookup functions... :

- The sort procedure was changed to gain speed. It is a hybrid of quicksort/insertion sort implementation that perfoms quite well. This procedure is also used to build the in-memory indexes.
- Sort can be used with calculated and lookupFields when dataset CalcDisplayedRecordsOnly or LookUpDisplayedRecordsOnly are set to true (default)
- In-memory indexes provide fast binary search. They are used in locate, lookup, lookupfields...etc... operations. The In-memory indexes can be set at designtime with a new dialog (TsqlitePassDatabase.Indexed and TsqlitePassDatabase.Indexes properties)
- Locate, lookup and lookupFields implementation have been completely rewritten. These functions use extensively the TsqlitePassFieldFilters, TsqlitePassFieldFilter and TsqlitePassFieldFilterExp classes.
- Support for BCD fields (up to 4 digits) is implemented. BCD fields are stored as int64 in the database.
- Bug correction with unicode support that could prevent from opening database when database path was too long or included unwanted chars,
- Internal recordset buffer implementation was changed again to provide more compact memory use and faster records management.
- Modifications in filters, locate dialogs,
- New Lookup dialog in SqliteToolbox demo program.
- Experimental Sqlite User Defined Functions support
- Help file partially updated.
- And many other bugfixes and improvements....

08/10/2009 – Version 0.45

The major changes are the components compatibilty with Delphi2009, Unicode support and better performances when loading and writing data. Here is a list of the most important changes :

- Rework on TSqlitePassDatabase.Open (function SqlitePassDatabase.CheckCanOpen). When SmartOpen is set to True :
  Two databases with the same filename but located in two different directories could lead to open the wrong database. Now a message will warn you.

- Rework on TSqlitePassDatabaseDataTypeOptions.LoadFromDatabase to stop raising uncessary exception when a database is opened for the first time

- Rework on "Internal" record Filters
  Filter functions are now defined as "inline" to speed up comparison
  ftString Fields were not correctly filetered. Now the text filter uses AnsiStrComp,

AnsiStrIComp, AnsiStrLComp and AnsiStrPos to perform comparisons.

- Like and NotLike were removed from internal filter syntax. Use '=' or '<>' instead with '*' and '%' Wildcard chars if needed

- TSqlitePassRecordset was modified to speed up loading data from SQL statements and to correct memory use that could lead to critical errors with previous version. The loading time is now about 10 times faster then version 0.42 and large tables or queries are now supported.

- Procedure TSqlitePassSQLTokenizer.Clean was rewritten

- TSqlitePassDataset.Refresh is implemented and works correctly now.

- Database encoding is now set correctly when a new database is created (UTF8 ,UTF16 , UTF16le, UTF16be)
-
- Database pagesize is now set correctly when a new database is created (512..32768)
-
- **Unicode UTF8 and UTF16 support :**

  The components should be 'unicode friendly' with UTF8 and UTF16 support for SQL statements, Table names, field names, field Data...etc.

  The property UnicodeEncoding sets how text, strings or memo are retrieved from the database, encoded, displayed and eventually put back to the database.

  This property is independant of the Database.Options.Encoding property which is readonly :
  you can read or write strings as UTF16 from an UTF8 Database and vice-versa. (To be tested...)

  UnicodeEncoding can take one of the following value (ueAuto, ueUTF8, ueUTF16, ueRawText).
  UnicodeEncoding doesn't really change the way text is encoded. It simply overwrite the field definition for a given table column, changing the original ftString to ftWideString, ftMemo to ftWideMemo or vice-versa.

  Whatever delphi or fpc version you use (unless you set UnicodeEncoding to ueRawText), SqlitePassDatabase uses the following rules :

  - A ftString field is ALWAYS encoded as UTF8,
  - A ftMemo field is ALWAYS encoded as UTF8,
  - A ftWideString field is ALWAYS encoded as UTF16,
  - A ftWideMemo field is ALWAYS encoded as UTF16.)

  The fields conversions applies only for dynamically created fields (not the ones already set at design time).
  If a custom conversion rule is already set for a given table field, it will always have the priority for the final field definition. See also Database.DatatypeOptions.CustomFieldDefs).

  The UnicodeEncoding default value is ueAuto.

  - ueAuto: no change is done to the fieldDefs.
  - ueUtf8: ftWideString fields are converted to ftString, ftWideMemo are converted to ftMemo,
  - ueUtf16: ftString fields are converted to ftWideString, ftMemo are converted to ftWideMemo,
  - ueRawText: no change is done to the fieldDefs, no encoding is done when retrieving data.)

Note :
Delphi, prior Delphi 2009, doesn't handle Unicode in classic vcl. In order to write Unicode applications you have to use third part libraries like tnt unicode controls or utf8-vcl (freeware). The delphi 4 demo program uses utf8-vcl which is not very reliable (memory leaks and pointers crashes) but good enougth to test the database components.

Delphi 2009 offers native Unicode support as UTF16.
Lazarus-fpc offers native Unicode support as UTF8 : it is a nice alternative to Delphi prior Delphi 2009 to build unicode applications.

- Many other changes are included in this version

10/05/2009 – Version 0.42

- Dataset.Filters were entirely rewritten to enable filtering on calculated fields with the addition of a new filter form. It is the most important addition to this version.
- Functions added to TsqlitePassDatabaseDataTypeOptions used to get a text representation of the Date Value, using the DateFormat property
- + function DateTimeToStr(Value: TDateTime): String;
- + function DateToStr(Value: TDateTime): String;
- + function TimeToStr(Value: TDateTime): String;
- A new component is now registered in IDE : TsqliteDBActionList. Drop this component on your form and set the datasource property. DoubleClic to add some of the preset available SqlitePassActions as you do with clasic ActionList editor. Link these DbActions to buttons or menus and you will be able to easely sort, filter, locate data etc... with dedicated forms. You can check this in the demo program as it uses this new component.
- TsqliteRecordset seems to be quite reliable now with many bugs fixes and some functions added. Deleted records are now recycled and Blobs get back to there prior state when editing is canceled.
- A lot of work was also done on TsqlitePassSQLTokenizer
- **Bugfix :** Inserting a record after a delete was buggy. Special thanks to Parcel who send this bug report and many bug fixes on forum.
- Demo program update.
- Help partially updated
- and many other minor changes...

10/01/2009 – Version 0.41

- This version introduces changes in the TsqlitePassSelectStmt object , with a new SQL Tokenizer (TsqlitePassSQLTokenizer - still in progress) and a dedicated SQL sections splitter (TsqlitePassSQLSections). Additional modifications of code when posting data (Insert-Update-Delete) has be done too.
- Added a specific TSqlitePassSQLStmtDef object to store SQLStmts directly in the database (TSqlitePassSQLStmtDefs) . The TSqlitePassSQLStmt object is now used when the statement need to be modified or send to the database engine.
- Better support for AutoInc Fields and PrimaryKey Fields.
- SQL functions like sum() are now supported (to be improved),
- Rework has been done on attached databases. All the database parts collections have an « Assign » procedure.
- **Bugfix :** in Date/DateTime conversion from databases created with SQLiteExpert. Thanks to Dsutuno for reporting this problem.
- **Bugfix :** in Function TSqlitePassRecordset.GetRowId(RecordIndex: Integer; TableNo: Word): Int64; that could return a incorrect RowId and update the wrong data row...
- Added Int64 support in the TSqlitePassDataset.sort procedure.
- Demo program update.

08/12/2008 – Version 0.40

- This version introduces too many changes to notice all of them.
- The TSqlitePassRecordset object was completely re-written and many parts of the dataset component had to be modified. These changes should lead to faster data reading and writing.
- Null values are handled
- The TSqlitePassTranslator object was completely re-written too, in order to add better support for date/time/datetime/boolean fields
- You will find a new documentation (no finished...) using PasDoc
- **Bugfix :** Procedure TsqlitePassDataset.FindFirst : was always returning True even when no record was found. (Thanks to John MORRIS for reporting this problem)
- Sorting was completely re-written and uses as default choice an internal QuickSort algorithm that should handle date/time, Memo and other datatypes correctly,
- and many others...

Many thanks to all of you who spend time to try these components and their bugs...

17/06/2008 – Version 0.35

Here are some of the main changes for this version :

- **Bugfix :** Binding of ftAutoinc fileds was not correct.
- **Bugfix :** Procedure TsqlitePassDataset.Locate : was always returning True even when no record was found.
- Added new collection TsqlitePassSQLStmts to store SQL statements directly into the database,
- Rework on TSQlitePassDataset.Post to improve post errors management,
- Demo program improvements :
    - You can now create new databases.
    - Tables can also be created, modified (partially), deleted, renamed, reindexed...etc
    - Rework on Database Options Dialogs (CustomfieldDefs and Mapping Rules)
    - Added several PopupMenus.
    - SQL tabsheet modified...etc
    - New installer for windows.

- New procedure CopyTable to duplicate a table structure with,or without its data
- And others small changes...

23/03/2008 – Version 0.34

Here are some of the main changes for this version :

- Dataset.Locate implementation, with
  New methods :
    - Locate
    - LocateFirst
    - LocatePrior
    - LocateNext
    - LocateLast

  New properties
    - LocateRecordCount
    - LocateMoveState
    - LocateSmartRefresh

- Dataset.Lookup implementation, with new methods :
    - Lookup
    - LookupFirst

    - LookupPrior
    - LookupNext
    - LookupLast

  New property
    - LookupSmartRefresh (Not implemented yet)

- Lookup fields are now supported.
- New TsqlitePassDatabaseError object to log and retrieve errors.

```
...
procedure GetLastError(var ErrorCode: Integer;var Msg: String;
var Time: TDateTime);
procedure SaveToFile(FileName: String);
property ErrorCount: Integer Read FErrorCount;
property ErrorList: TStringList Read FErrorStringList;
```

TsqlitePassDatabaseOptions.LogErrors property

- Many bugs corrections and code changes,
- **Demo program has been updated to help you in testing and using the new features.**

29/02/2008 – Version 0.33

Here are some of the main changes for this version :

- Rework on filtering dataset data. I forgot to post the changes in version 0.32...It should work correctly now,
- Dataset FindFirst, FindLast, FindNext, FindPrior are implementation for filtered datasets
- Dataset Fields can now be defined at designtime with the IDE fields editor
- Rework on Dataset.InternalOpen, InternalInitFieldDefs and new utility function « CheckCanOpen ».
- New Filter editor at designtime
- **Bugfix :** Procedure TsqlitePassDatabase.SaveToDatabase and procedure TSqlitePassTableDefs.Refresh;
  Thanks to Leander007 for his contribution (details can be found on sqlitepass forum)
- New TsqlitePassDatabase.CreateDatabase procedure
  Thanks to chukkan for his contribution (details can be found on sqlitepass forum)
- New installation packages for Delphi 6 and 7
- And many other changes...

03/02/2008 – Version 0.32

This is the first version running under linux (ubuntu). It should compile quite correctly. Let me know if you have problems.

Here are some of the main changes for this version :

- Rework on filtering dataset data. It should work correctly now,
- Rework on Blob datafield to fix memory leaks, speed up blob display and save memory. (I still have problem under lazarus to display images using TdbImage component)
- TsqlitePassDatabase should now be able to open/read/write to databases created with kexi, Sqlite Expert, Sqlite Admin or Sqlite4Fpc. It should also be able to open any other sqlite databases if you define the right parameters in the Database.DataTypeOptions properties :

  Property BooleanExtension
  Property CustomFieldDefs
  Property DefaultFieldType
  Property DetectionMode
  Property DateSeparator
  Property DateStorage;
  Property LoadOptions
  Property LongTimeFormat
  Property ShortDateFormat
  Property SaveOptions
  Property TimeSeparator
  Property TranslationRules

- New property Database.Options : QuoteStyle, used to quote fieldnames containing spaces,
- Updated help file,
- And many other small improvements or corrections.

04/01/2008 – Version 0.31

This version introduces the basic for futur locate / lookup functions and some bugfixes. I still had some memory leaks to fix from the previous versions. I used Fastmm from http://fastmm.sourceforge.net to test memory allocation/deallocation and to catch pointer missuse while using Delphi 4.
Many thanks to Pierre le Riche for this tool.

Here are some of the main changes for this version :

- **Bugfix :** Procedure TsqlitePassDataset.InternalOpen;
  Memory leak fix. FSQLSelectStmt was created twice (in TsqlitePassDataset.Create and TsqlitePassDataset.InternalOpen and fred just once...)
- Improved support for attached databases : Attached tables return correct record data when several attached tables have the same name.
- TsqlitePassDatabase.refreshDefinitions and underlying translator implementations was rewritten to improve attached database support.
- Added basic Triggers support.
- And many small improvements or corrections

12/09/2007 – Version 0.30

This version introduces Master/Detail filters and a lot of bugfixes. Many memory leaks had to be fixed from the previous versions. I used the fpc heaptrc.pas unit to test memory allocation/deallocation. This unit is a wonderful tool to catch memory errors. (In lazarus IDE, compiler options, Linker, check « use heaptrc ».

Here are some of the main changes for this version :

- **Bugfix :** Function TSqlitePassDatabase.GetFCollatingOrder:String; Wasn't returning any result...It now returns a default empty string as a result and doesn't raise a runtime error anymore.
- **Bugfix :** TsqlitePassDataset.DatasetName was set to '' (empty) when SQL property was set, even if the SQL text remained the same as before.
- **Bugfix :** TsqlitePassDataset.Delete didn't call the right SQL statement to delete record. Should be fixed now.
- **Bugfix :** Sqlite library was always unloaded when engine destroyed even if more than one engine was using it. It now keeps track of the different engines running (counter) and the library is released only when the last engine is destroyed.
- **Bugfixes :** Many objects constructors/destructors have been rewritten to avoid runtime errors,
- New procedure TsqlitePassMasterDataLink.LayoutChanged to synchronise internal Recordset fields order with Dataset fields,
- New properties : MasterSource, MasterFields, MasterAutoActivate,...
- New Event: OnFilterRecord to enable custom filter (Blob fields are available when filtering),
- New property editor to link Master/Detail fields,
- procedure TsqlitePassDataset.InternalRefresh was rewritten to speed up filetring,
- TsqlitePassSelectStmt was partially rewritten to give a more reliable schema structure,
- Help file have been partially updated
- ...etc

19/07/2007 – Version 0.29

- **Bugfix** for Procedure TsqlitePassDatabase.SetFSQLStmt , TsqlitePassDataset.SetFDatasetType, TsqlitePassDataset.SetFDatasetName : modification to avoid problems when changing sql property at designtime. Thanks to Wilhelm Reeg for reporting this problem.
- **Bugfixes** in the indexDefs.Refresh property (indexdefs was not retreived)
- Improvement in the demo program (It displays info on FieldDefs now...)
- Added support for filtering and improvemnt in sorting editor.
- Delphi version of packages will compile directly to c:\windows\system32 now. Change the outpout directory in package options if your configuration is different.

28/06/2007 – Version 0.28

- It uses now the sqlitepass3.dll as the default sqlite library (this library exports the following functions to get schema information from an sql statement : These functions are :
  - *sqlite3_column_database_name*
  - *sqlite3_column_database_name16*
  - *sqlite3_column_table_name*
  - *sqlite3_column_table_name16*
  - *sqlite3_column_origin_name*
  - *sqlite3_column_origin_name16*
  - *sqlite3_table_column_metadata*

- Calculated fields are implemented.
- **Bugfix** for Procedure TsqlitePassDatabase.SetFSQLStmt and TsqlitePassDataset.SetFDatasetType : modification to handle calculated fields
- **Bugfix** for Procedure TsqlitePassDatabase.Loaded : « inherited loaded ». It is correct now.
- SqlitePassDbo contains only an interface section now. All implementations have been splitted and move to *.inc files. This offers easier coding (objects can use each other without writting an interface definition or caring about unit circular references...)
- TsqlitePassFieldDefs is used now to retrieve fielddefs information from database. This is used by TSqlitePassSelectStmt to extract a shema of the fields used in the dataset select statement. This information is used to be able to write from a query that use several tables. Yes, you can update queries now !
- Many other changes have been made to units structure and source code to keep track of every modifications. The basic overall component structure should be almost done (even if implementation is sometime missing)
- The documentation was not updated. Since the code is changing to quickly, I will write it one the components are enough tested...

28/04/2007 – Version 0.27

- TsqlitePassDatasets.FindDataset is implemented.
- TSqlitePassDatasets.Close is implemented.
- TSqlitePassRecordsetCache was heavely modified again. Many bugs fixed and speed improvements. The change made in version 0.26 « *When adding a new record, TsqlitePassRecordsetCache doesn't use a special record anymore but create a new record in the global records list. This record is removed by the dataset if finally the user cancels the insert/append operation* » was deleted. I use a new record again (FnewRecord) because the previous changes was too confusing with the count property and when navigating in the recordset.
- InitFieldDefs is moved to TSqlitePassTranslator to enable better fieldtype detection with Kexi.
- **Bugfix** Procedure TsqlitePassDatabase.SetFActive : Factive was not set at all...This is now correctly set.

- **Bugfix for Procedure TsqlitePassDataset.SetFDatasetName : Dataset Name was not correctly retrieved from *.lfm or *.dfm when set at designtime and no data was displayed in application at runtime.**
- **Bugfix** for Procedure TsqlitePassDataset.GetRecord : No data was retrieve when DatasetState was dsInactive
- Added support for int64 (ftLargeInt) field type in TsqlitePassTranslator.
- FieldDefs has been completed for Kexi Databases, especially for fields constraints. The TsqlitePassBitArray was created to take care of this (SqlitePassParser Unit).
- **Bugfix** when adding a record, recordcount was always updated. It is now updated only if the new record is really posted.
- When inserting or appending a new record, Procedure TsqlitePassDataset.SetBookmarkFlag and TsqlitePassDataset.SetBookmarkData store the current position now. This enables to go back to the last selected record if inserting is canceled.
- And more...

17/03/2007 – Version 0.26

- Code cleanup has been done, especially in SqlitPassDbo unit. Lots of « TODO » lists are left...
- Blobs support is now implemented and seems to work quite properly in Delphi but Lazarus doesn't display picture in TdbImage component...
- Packages and components versions have been changed to provide a better version installation check in IDE-object inspector,
- New documentation file. Basic information on packages contents and installation for now...
- Heavy changes and cleanup in TsqlitePassRecordsetCache :
    - Old fields values are saved now and can be restored if needed,
    - When adding a new record, TsqlitePassRecordsetCache doesn't use a special record anymore but create a new record in the global records list. This record is removed by the dataset if finally the user cancels the insert/append operation. Procedure TsqlitePassDataset.InternalCancel is implemented.
    - GetRecords methods has been completely rewritten to add a LowerRecordLimit and an UpperRecordLimit filter. More changes are planed, especially in fields type detection.
- The sqlite library file to be used can now be selected directly in object inspector. If no file is selected, the default one will be used,
- SqlitePassDataset uses only a TMemRecord pointer value when sending or retrieving information to/from TDataset. This should really speed up operations since no real data is moved in memory,
- New indexes designtime editor,
- New Sqlite library designtime editor,
- Unit SqlitePassApi_v3 : The procedure 'LoadFunctions' had to be rewritten to enable dynamic loading of the library (.dll or .so) by the TSqlitePassDatabase.SQLiteLibrary property;
- And more...

20/01/2007 – Version 0.25

- Procedure TSqlitePassDataset.SetFieldData(Field: TField; Buffer: Pointer); virtual; abstract;
  The DataEvent is used to set TDataset.SetModified to True.
  Then TDataset.CheckBrowseMode will be able to Post the record if necessary when scrolling the dataset

  ```
  if not (State in [dsInternalCalc, dsCalcFields, dsFilter, dsNewValue])
    then DataEvent(deFieldChange, Longint(Field));
  ```

- The procedure TSqlitePassIndex.SetFIndexName(Value: String) checks the index name to be set (same rules as a pascal identifier) to avoid error when renaming or deleting index.

if IsValidIdent(Value)

- Added SQLPassParser unit to enable easier data sorting, filtering in TSqlitePassDataset
- New sortby designtime editor
- Many bug fixes and changes.

18/12/2006 – Version 0.24

- Added TsqlitePassDatabasesAttached object to maintain a list of external databases attached to the current database.
  - *Attached Tables support*
  - *Attached Queries support*
- Rework on date/time support for lazarus since Tdataset.GetFieldData implementation has changed with FPC 2.0.4 / Lazarus 0.9.18

17/10/2006 – Version 0.23

- Added TSqlitePassTranslator.GetIndexDefs;
- IndexDefs[i].Sql corrected to FDatabase.IndexDefs[i].Sql
- Added new procedure CreateIndex(Index: TSqlitePassDatasetIndex); overload;
- Added basic TSqlitePassViews implementation and Database.Views property
- Changes made in TSqliteDataset.internalOpen to handle views.

24.09.2006 – Version 0.22

- TsqlitePassTranslator_Kexi.GetDatabaseObjects : the SQL statement is correct now.
- TsqlitePassDataset.InternalInitFieldDefs uses now the new TSqlitePassRecordsetCache.HasPrimaryKey property to return the correct fieldDefs.
- TSqliteIndexDefs - TSqliteIndex - TSqliteIndexColumns – TsqliteIndexColumn Partially implemented
- Added TSqlitePassIndexColumn.Position property

03.09.2006 – Version 0.21

- Improved date/time support for lazarus, some bugs fixed with queries, fielddefs...
- Added partial support for Kexi system tables in TsqlitePassTranslator_Kexi and more...

20.07.2006 – Version 0.20 - First public release. For testing and debuging...

Known Bugs

This is a list of known bugs in the last package version. Feel free to report new bugs, send bugs fixes to SqlitePass Forum at http://source.online.free.fr/

Lazarus

- TdbImage component doesn't display any picture in Demo program...
- Locate raises SIGV error on cancel

Delphi

- No heavy bugs found – Please report...

TODO List

This is a list of TO DO for the next releases :

General

- Improve write autorisation on queries
- Rework on Triggers,
- Add data Importing, Exporting
- Add table modification without loosing data
- Add possibility of copying or moving Tables data or structure, Queries...etc between databases or attached databases,
- Add better support for cloned fields in SQLStmt,
- Add better support for autoinc and primarykey fields in SQLStmt,
- Continu the help file,
- Add profiler to test library speed,
- etc...

Lazarus

- Fix TdbImage problem

Delphi