



А.В. Лисицын
М.В. Лисицын
К.С. Гудков

« wxLib »
кроссплатформенная
инструментальная библиотека
для разработки приложений баз данных
в среде *Lazarus & FPC*

*** * * свободное ПО с открытым исходным кодом * * ***



Москва 2026



«wxLib» для

<https://ruSBSS.ru>



ПРЕДЫСТОРИЯ

Инструментальную библиотеку «wxLib» авторы начали разрабатывать в 1998г. Изначально она предназначалась для унификации процесса разработки Windows-приложений баз данных в среде программирования Delphi. С тех пор библиотека «wxLib» позволила нам сэкономить массу сил и времени при разработке целого ряда информационных систем.

В силу известной тенденции к обеспечению импортонезависимости в российской IT-отрасли, обусловившей необходимость перехода на отечественное, свободное и открытое ПО, в 2018г мы приступили к портированию библиотеки в среду Lazarus + FPC (Free Pascal Compiler).

В данном руководстве представлено описание библиотеки по состоянию на июнь 2024г.

Исходные коды «wxLib» представлены в редакции 2026-06-18.

Библиотека и сопутствующий ей инструментарий (утилиты) позволяют эффективно разрабатывать надёжные кроссплатформенные приложения баз данных для операционных сред Microsoft Windows и Linux. При этом доступ к базам данных реализован в ядре библиотеки на мультисерверном подходе и поддерживает 15 типов различных СУБД.

Библиотека «wxLib» является свободным программным продуктом с открытым исходным кодом и распространяется по лицензии LGPL.

БЛАГОДАРНОСТИ

Авторы выражают глубокую признательность своим коллегам, А.С.Голубеву и С.Е.Арабаджиеву за реализацию компонент генерации отчётности в модулях «wxLib» и А.Н.Потёмкину за разработку и поддержку сайта <https://rusbss.ru>.

ОБ АВТОРАХ

Лисицын Андрей Владимирович



Образование: Московский институт электроники и математики (МИЭМ). Кандидат технических наук. Опыт профессиональной работы в области информационных технологий – с 1975г. Области профессиональных интересов: прикладная математика, теплофизика, информационные технологии, инструментальные средства программирования и прототипирования кроссплатформенных приложений, системы управления базами данных, кроссплатформенные распределённые информационные системы, системы репликации баз данных.

Лисицын Михаил Владимирович



Образование: Московский институт электроники и математики (МИЭМ). Опыт профессиональной работы в области информационных технологий – с 1985г. Области профессиональных интересов: информационные технологии, инструментальные средства программирования и прототипирования кроссплатформенных приложений, системы управления базами данных, кроссплатформенные распределённые информационные системы, системы репликации баз данных.

Гудков Кирилл Сергеевич



Образование: Московский физико-технический институт (МФТИ). Кандидат физико-математических наук. Опыт профессиональной работы в области информационных технологий – с 2003г. Области профессиональных интересов: прикладная математика, математическое моделирование, информационные технологии, системы управления базами данных, системы репликации баз данных.

ОГЛАВЛЕНИЕ

1.	Общая информация о библиотеке <i>WXL</i>	7
1.1	Назначение библиотеки <i>WXL</i>	7
1.2	Структура библиотеки <i>WXL</i>	7
1.3	Установка библиотеки <i>WXL</i>	20
1.4	Настройка файла <i>wxdefs.inc</i>	21
2.	Используемые компоненты в составе <i>Lazarus</i>	26
2.1	<i>SQLdb</i>	26
2.2	<i>SynEdit</i>	27
2.3	<i>LazReport</i>	29
2.4	<i>TDBF</i>	32
3.	Связные библиотеки	34
3.1	<i>ZeosDBO</i>	34
3.2	<i>UniDAC</i>	36
3.3	<i>FastReport</i>	38
4.	<i>WXL</i> -утилиты	43
4.1	Утилита <i>wxlgen</i>	43
4.1.1	Что первично в программировании приложений баз данных	43
4.1.2	Описание генератора <i>wxlgen</i>	44
4.1.3	Генерация SQL-скриптов для объектов БД	59
4.1.4	Генерация исходных текстов и ресурсов <i>wxl</i> -приложений	60
4.1.5	Структура типового <i>wxl</i> -приложения	65
4.2	Утилита <i>wxldbconmgr</i>	75
4.2.1	Описание утилиты	75
4.2.2	Использование файлов <i>wxldbcon_*.lst</i> в <i>wxl</i> -приложениях	78
4.3	Утилита <i>wxetl</i>	79
5.	Основы графического интерфейса <i>wxl</i> -приложений	87
6.	Доступ к данным	98
6.1	Технологии доступа к данным из приложения	98
6.2	Проблема пустых данных (<i>NULL</i>)	103
6.3	Наборы данных в памяти и компонент <i>TWxMemDataSet</i>	106
6.4	Модули доступа к данным библиотеки <i>wxl</i>	107
7.	Манипулирование данными	110
7.1	Компоненты ввода	110
7.1.1	Однострочные компоненты для ввода строк	110
7.1.2	Компоненты для ввода целых чисел	111
7.1.3	Компоненты для ввода вещественных чисел	113
7.1.4	Компоненты для ввода дат и времени	114

7.1.5	TWxEntryBoolean	116
7.1.6	Многострочные компоненты для ввода строк	116
7.2	Сетки	117
7.2.1	TWxGrid	117
7.3	Справочники	118
7.3.1	TWxLinkButton	119
7.3.2	TWxDictButton	120
7.3.3	TWxDictSource	121
7.4	Дополнительные кнопки	123
7.4.1	TWxBitBtn	123
7.4.2	TWxSpeedButton	123
7.5	Особенности использования компонента <i>TWxTransfer</i>	124
7.6	Манипулирование данными в <i>wxl</i> -приложениях	125
7.7	GUI-модули для манипулирования данными	127
7.8	Модули без GUI для манипулирования данными	128
8.	Генерация отчётности	130
8.1	Общая информация	130
8.2	Модули для работы с отчётностью	133
9.	Модули для работы с ОС	134
10.	Взаимодействие с СУБД	136
10.1	Поддерживаемые СУБД и модуль <i>wxtypes</i>	136
10.2	GUI-модули для взаимодействия с различными СУБД	136
10.3	Модули без <i>GUI</i> для взаимодействия с различными СУБД	138
10.4	<i>FrmSQLEx</i> и встроенная среда разработки на языке <i>SQL</i>	139
10.5	Сохранение и восстановление БД	141
10.5.1	Общие подходы	141
10.5.2	Особенности, накладываемые системами репликации	143
10.5.3	Средства выполнения <i>Backup</i> в <i>WXL</i> -приложениях	146
10.6	Генерация уникальных идентификаторов	148
10.7	Блокировка записей в таблицах БД	152
11.	Работа с пользователями	157
11.1	Многопользовательский доступ к данным	157
11.2	Модули для работы с пользователями <i>wxl</i> -приложений	162
12.	Прочие модули библиотеки <i>WXL</i>	163

1. Общая информация о библиотеке WXL

1.1 Назначение библиотеки WXL

Назначение библиотеки WXL – по возможности облегчить программирование приложений, работающих с базами данных. Доступ к данным осуществляется унифицированным образом, вне зависимости от используемой технологии доступа к данным. Единая структура библиотеки позволяет автоматически генерировать шаблоны для WXL-приложений. Компоненты библиотеки предоставляют широчайший выбор возможностей, которые Вы можете использовать в своих приложениях.

1.2 Структура библиотеки WXL

Модули и компоненты библиотеки можно условно поделить на семь групп. Дадим краткую характеристику каждой из них.

- 1) Модули и компоненты для доступа к данным.
 - 1.1) Специфичные для технологии доступа к данным *SQLdb* (размещены в подпапке *sqldb* исходников библиотеки WXL).
 - 1.1.1) Модуль *WxDBDecimal* содержит функцию *ScaledInt64ToBcd* для преобразования *Int64* в *BCD* с указанием числа знаков после запятой (масштаб, *scale*).
 - 1.1.2) Модуль *WxDBSQL_FPC* обеспечивает доступ к *SQL*-серверам с использованием технологии доступа к данным *SQLDb*.
 - 1.1.3) Модуль *WxDBSQL_FPCUtils* содержит реализацию режима отладки для выполняемых *SQL*-запросов с использованием технологии доступа к данным *SQLDb*.
 - 1.1.4) Модуль *WxODBC* является вспомогательным для модуля *WxODBCConn*.
 - 1.1.5) Модуль *WxODBCConn* является вспомогательным для модуля *WxDBSQL_FPC* для обеспечения работы с *SQL*-серверами через *ODBC* с использованием технологии доступа к данным *SQLDb*.
 - 1.1.6) Модуль *WxODBCSQL* является вспомогательным для модулей *WxODBC* и *WxODBCConn*.

- 1.2) Специфичные для технологии доступа к данным *UniDAC* (размещены в подпапке *unidac* исходников библиотеки *WXL*).
- 1.2.1) Модуль *WxDbSQL_UNI* обеспечивает доступ к *SQL*-серверам с использованием технологии доступа к данным *UNIDAC*.
- 1.2.2) Модуль *WxDbSQL_UNIUtils* содержит реализацию режима отладки для выполняемых *SQL*-запросов с использованием технологии доступа к данным *UNIDAC*.
- 1.2.3) Модуль *WxUniMapRuleDlg* – диалог для сопоставления поля типу данных.
- 1.2.4) Модуль *WxUniMapRuleFra* – фрейм для сопоставления поля типу данных.
- 1.2.5) Модуль *WxUniMapRulesDlg* – диалог для сопоставления полей и типов данных.
- 1.2.6) Модуль *WxUniMapRulesFra* – фрейм для сопоставления полей и типов данных.
- 1.2.7) Модуль *WxUniSpecificOptionsDlg* – диалог для задания расширенных настроек соединения *UniDAC* (*SpecificOptions*).
- 1.2.8) Модуль *WxUniSpecificOptionsFra* – фрейм для задания расширенных настроек соединения *UniDAC* (*SpecificOptions*).
- 1.2.9) Модуль *WxUniTypes* – вспомогательный модуль для модулей сопоставления полей и типов данных: *WxUniMapRuleDlg*, *WxUniMapRuleFra*, *WxUniMapRulesDlg* и *WxUniMapRulesFra*.
- 1.3) Специфичные для технологии доступа к данным *ZeosDBO* (размещены в подпапке *zeos* исходников библиотеки *WXL*).
- 1.3.1) Модуль *WxDbSQL_ZEOS* обеспечивает доступ к *SQL*-серверам с использованием технологии доступа к данным *ZeosDBO*.
- 1.3.2) Модуль *WxDbSQL_ZeosUtils* содержит реализацию режима отладки для выполняемых *SQL*-запросов с использованием технологии доступа к данным *ZeosDBO*.
- 1.4) Прочие *GUI*-модули для доступа к данным (размещены в подпапке *gui* исходников библиотеки *WXL*).
- 1.4.1) Модуль *WxAsyncQuery* содержит реализацию класса для асинхронного выполнения *SQL*-запроса.

- 1.4.2) Модуль *WxDBSQLConnectDlg* содержит реализацию диалога для соединения с СУБД.
- 1.4.3) Модуль *WxDBSQLConnectFrm* содержит реализацию формы для соединения с выбранной СУБД (в частности, используется в генераторе *wxlgen*).
- 1.4.4) Модуль *WxDBSQLConnectList* содержит реализацию набора данных для хранения строк соединения на основе *TWxMemDataSet*.
- 1.4.5) Модуль *WxDBSQLConnectListFrm* содержит форму для выполнения выбранных действий над набором данных для хранения строк соединения: добавление соединения, исправление соединения, удаление соединения, проверка соединения.
- 1.4.6) Модуль *WxDBSQLConnectPanel* содержит реализацию вспомогательного класса *TWxDBSQLConnectFrame* для модуля *WxDBSQLConnectDlg* на основе панели (*TPanel*).
- 1.5) Прочие модули без *GUI* для доступа к данным.
- 1.5.1) Модуль *WxConnectString* содержит функциональность по построению строки соединения.
- 1.5.2) Модуль *WxDBISAM* обеспечивает функциональность для работы с локальными базами данных.
- 1.5.3) Модуль *WxDBParams* предназначен для поддержки работы с параметрами *TParam*.
- 1.5.4) Модуль *WxDBSQL* обеспечивает доступ к *SQL*-серверам. Использует функциональность одного из модулей *WxDBSQL_FPC*, *WxDBSQL_UNI* или *WxDBSQL_Zeos* в зависимости от выбранных директив условной компиляции.
- 1.5.5) Модуль *WxDBSQLProxy* является вспомогательным для модуля *WxDBSQLConnectPanel*.
- 1.5.6) Модуль *WxMemDataSet* содержит реализацию набора данных в виде таблицы в памяти *TWxMemDataSet*, который не связан с какой-либо базой данных.
- 1.6) Компонент *TWxMemDataSet* представляет собой набор данных в виде таблицы в памяти, который не связан с какой-либо конкретной базой данных. В документации, касающейся данного компонента, имеется пример его использования (раздел 6.3).

- 2) Модули и компоненты для манипулирования данными.
 - 2.1) *GUI*-модули для манипулирования данными (размещены в подпапке *gui* исходников библиотеки *WXL*).
 - 2.1.1) Модуль *WxButtons* содержит реализацию *WXL*-компонентов для кнопок: *TWxBitBtn*, *TWxSpeedButton*, *TWxLinkButton*.
 - 2.1.2) Модуль *WxDataSetExportFrm* содержит реализацию формы для экспорта набора данных в файл.
 - 2.1.3) Модуль *WxDataSetSort* содержит классы и функции, ориентированные на сортировку извлечённых из базы данных таблиц по заданным пользователем столбцам. Используется модулем *WxDbGrid*.
 - 2.1.4) Модуль *WxDataSetView* содержит класс и функции, ориентированные на просмотр таблиц баз данных.
 - 2.1.5) Модуль *WxEntry* содержит классы и функции, предназначенные для реализации компонентов ввода-вывода данных.
 - 2.1.6) Модуль *WxExportParamsPanelCSV* содержит реализацию панели для выбора параметров экспорта в *CSV*-формате. Является вспомогательным для модуля *WxDataSetExportFrm*.
 - 2.1.7) Модуль *WxExportParamsPanelSQL* содержит реализацию панели для выбора параметров экспорта в *SQL*-формате. Является вспомогательным для модуля *WxDataSetExportFrm*.
 - 2.1.8) Модуль *WxTblDict* содержит классы и функции, ориентированные на работу со словарём.
 - 2.1.9) Модуль *WxDbGrid* содержит реализацию визуального компонента *WxDbGrid*.
 - 2.1.10) Модуль *WxDbGridDlg* является вспомогательным для модуля *WxDbGrid*. В нём содержится реализация опций, вызываемых по всплывающему *PopupMenu WxGrid* по умолчанию.
 - 2.2) Модули без *GUI* для манипулирования данными.
 - 2.2.1) Модуль *WxDataSet* содержит функциональность по созданию различных наборов данных в памяти на основе *TDataSet*.
 - 2.2.2) Модуль *WxDataSet2BDS* содержит функциональность по конвертации набора данных *TDataSet* в специальный бинарный *BDS*-формат.

- 2.2.3) Модуль *WxDataSet2KBM* содержит функциональность по конвертации набора данных *TDataSet* в *KBM*-формат.
- 2.2.4) Модуль *WxDataSet2VTD* содержит функциональность по конвертации набора данных *TDataSet* в *VTD*-формат компонента *VirtualTable* в составе *UniDAC*.
- 2.2.5) Модуль *WxDataSetExport* является вспомогательным для модуля *WxDbGrid*. Он предназначен для экспорта набора данных в файл выбранного формата. Поддерживаемые форматы: *TXT, XLS, HTM, XML, ADT, DB, DBF, DAT*.
- 2.2.6) Модуль *WxDataSetInfo* предназначен для получения информации о наборе данных, используется как вспомогательный для модуля *WxDataSetView*.
- 2.2.7) Модуль *WxDataSetStorageBIN* предназначен для сохранения и загрузки набора данных в бинарном формате.
- 2.2.8) Модуль *WxDataSetStorageJSON* предназначен для сохранения и загрузки набора данных в формате *JSON*.
- 2.2.9) Модуль *WxDataSetStorageText* предназначен для сохранения и загрузки набора данных в текстовом формате.
- 2.2.10) Модуль *WxDataSetStorageXML* предназначен для сохранения и загрузки набора данных в формате *XML*.
- 2.2.11) Модуль *WxExportParams* содержит классы для параметров экспорта. Является вспомогательным для модулей *WxDataSetExport* и *WxDataSetExportFrm*.
- 2.2.12) Модуль *WxDataSetScan* является вспомогательным для модуля *WxDbGridDlg*. В нём содержится реализация средств для сканирования таблиц, которые затем используются для фильтрации и поиска записей таблицы, отвечающих набору условий по столбцам.
- 2.2.13) Модуль *WxDbCommon* содержит вспомогательную функциональность для модулей *WxDbGrid* и *WxDbGridDlg*.
- 2.3) Компоненты ввода данных
- 2.3.1) Компонент *TWxEntryShortString* предназначен для ввода строк типа *shortstring*.
- 2.3.2) Компонент *TWxEntryAnsiString* предназначен для ввода строк типа *AnsiString*.
- 2.3.3) Компонент *TWxEntryWideString* предназначен для ввода строк типа *widestring*.

- 2.3.4) Компонент *TWxEntryUnicodeString* предназначен для ввода строк типа *unicodestring*.
- 2.3.5) Компонент *TWxEntryInteger* предназначен для ввода целых чисел типа *integer*.
- 2.3.6) Компонент *TWxEntryLargeInt* предназначен для ввода целых чисел типа *largeint*.
- 2.3.7) Компонент *TWxEntrySmallInt* предназначен для ввода целых чисел типа *smallint*.
- 2.3.8) Компонент *TWxEntryShortInt* предназначен для ввода целых чисел типа *shortint*.
- 2.3.9) Компонент *TWxEntryWord* предназначен для ввода целых чисел типа *Word*.
- 2.3.10) Компонент *TWxEntryLongWord* предназначен для ввода целых чисел типа *LongWord*.
- 2.3.11) Компонент *TWxEntryByte* предназначен для ввода целых чисел типа *Byte*.
- 2.3.12) Компонент *TWxEntryCurrency* предназначен для ввода вещественных чисел типа *Currency*.
- 2.3.13) Компонент *TWxEntryFloat* предназначен для ввода вещественных чисел типа *Double*.
- 2.3.14) Компонент *TWxEntrySingle* предназначен для ввода вещественных чисел типа *Single*.
- 2.3.15) Компонент *TWxEntryExtended* предназначен для ввода вещественных чисел типа *Extended*.
- 2.3.16) Компонент *TWxEntryBCD* предназначен для ввода вещественных чисел типа *BCD*.
- 2.3.17) Компонент *TWxEntryDate* предназначен для ввода даты типа *TDateTime*.
- 2.3.18) Компонент *TWxEntryDateShortString* предназначен для ввода даты в виде строки типа *shortstring*.
- 2.3.19) Компонент *TWxEntryDateString* предназначен для ввода даты в виде строки типа *string*.
- 2.3.20) Компонент *TWxEntryTime* предназначен для ввода времени типа *TDateTime*.

- 2.3.21) Компонент *TWxEntryBoolean* предназначен для ввода логических значений типа *Boolean*.
- 2.3.22) Компонент *TWxMemoShortString* – *Memo*, использующий для хранения строк тип данных *ShortString*.
- 2.3.23) Компонент *TWxMemoAnsiString* – *Memo*, использующий для хранения строк тип данных *AnsiString*.
- 2.3.24) Компонент *TWxMemoWideString* – *Memo*, использующий для хранения строк тип данных *WideString*.
- 2.3.25) Компонент *TWxMemoUnicodeString* – *Memo*, использующий для хранения строк тип данных *UnicodeString*.
- 2.3.26) Компонент *TWxEntryString* предназначен для ввода строк типа *string*.
- 2.3.27) Компонент *TWxMemoString* – *Memo*, использующий для хранения строк тип данных *String*.
- 2.4) Сетка *TWxGrid* – многофункциональная сетка для отображения пользователю набора данных.
- 2.5) Справочники:
 - 2.5.1) Компонент *TWxDictButton* – кнопка вызова словаря.
 - 2.5.2) Компонент *TWxDictSource* – источник данных словаря.
 - 2.5.3) Компонент *TWxLinkButton* – кнопка, присоединённая к источнику получения-отображения данных.
- 2.6) Кнопки:
 - 2.6.1) Компонент *TWxBitBtn* – обычная кнопка с надписью или кнопка с выбранной картинкой и надписью.
 - 2.6.2) Компонент *TWxSpeedButton* – кнопка с картинкой, картинка может выбираться из списка стандартных *WXL*-картинок.
- 2.7) Компонент *WxTransfer* предназначен для объединения компонентов ввода-вывода в единую связанную структуру.
- 3) Модули для работы с отчётностью (размещены в подпапке *rpt* исходников библиотеки *WXL*).
 - 3.1) Модуль *DlgWxTblRpt* содержит реализацию формы для ввода параметров отчёта.

- 3.2) Модуль *WxTblRpt* ориентирован на создание отчётов *LazReport* или *FastReport* по набору данных.
- 3.3) Модуль *WxTblRptExp* – модуль данных для экспорта отчётов.
- 3.4) Модуль *WxTblRptFR6* ориентирован на создание отчётов по набору данных с использованием *FastReport*.
- 3.5) Модуль *WxTblRptFR6Exp* ориентирован на экспорт отчётов *FastReport*.
- 3.6) Модуль *WxTblRptL* ориентирован на создание отчётов по набору данных с использованием *LazReport*.
- 3.7) Модуль *WxTblRptLRexp* ориентирован на экспорт отчётов *LazReport*.
- 4) Модули для работы с операционной системой и файловой системой.
 - 4.1) *GUI*-модули для работы с операционной системой и файловой системой (размещены в подпапке *gui* исходников библиотеки *WXL*).
 - 4.1.1) Модуль *WxFileDialog* содержит диалоги для выбора файлов из файловой системы.
 - 4.1.2) Модуль *WxFileManagerDlg* содержит реализацию диалога для выполнения стандартных операций через файловый менеджер: копирование, перемещение, создание каталога, переименование и удаление.
 - 4.1.3) Модуль *WxFileManagerFra* содержит реализацию панели двухпанельного файлового менеджера.
 - 4.1.4) Модуль *WxFileManagerFrm* содержит реализацию основной формы двухпанельного файлового менеджера.
 - 4.1.5) Модуль *WxFileManagerInterface* содержит реализацию программного интерфейса для двухпанельного файлового менеджера.
 - 4.1.6) Модуль *WxFileManagerListView* содержит реализацию просмотрщика набора данных для хранения списка файлов на основе *TCustomDbGrid*.
 - 4.1.7) Модуль *WxTXTView* содержит реализацию формы просмотра текстовых файлов.
 - 4.2) Модули без *GUI* для работы с операционной системой и файловой системой.
 - 4.2.1) Модуль *WxClasses* содержит реализацию классов *TADFileStream* и *TStreamWriter*.

- 4.2.2) Модуль *WxFileAttr* содержит функциональность для работы с атрибутами файлов.
- 4.2.3) Модуль *WxFileManagerList* содержит реализацию набора данных на основе *TWxMemDataSet* для хранения списка файлов.
- 4.2.4) Модуль *WxFiles* содержит процедуры и функции для выполнения стандартных действий над файлами и папками в файловой системе (копирование, переименование, перемещение, ...).
- 4.2.5) Модуль *WxFileTime* содержит функции для просмотра и установки атрибутов даты создания, изменения и последнего доступа к файлам.
- 4.2.6) Модуль *WxFileUtils* содержит функциональность по сохранению и загрузке строк из потоков (*TStream*).
- 4.2.7) Модуль *WxFileVer* содержит функциональность по работе с версией продукта *Lazarus* (*Major, Minor, Release, Build*): получение версии продукта, сравнение версии продукта у двух заданных файлов.
- 4.2.8) Модуль *WxINIFiles* содержит функциональность по работе с *INI*-файлами.
- 4.2.9) Модуль *WxSysUtils* содержит несколько функций: получение имени компьютера, модуля и пользователя; получение доступной и используемой памяти.
- 5) Модули, предоставляющие функциональность *SQL*-сервера.
- 5.1) *GUI*-модули, предоставляющие функциональность *SQL*-сервера (размещены в подпапке *gui* исходников библиотеки *WXL*).
- 5.1.1) Модуль *FrmDbIndexes* служит для отображения индексов таблиц, принадлежащих БД. Является вспомогательным для модуля *FrmDbTables* через модуль *WxDBHelp*.
- 5.1.2) Модуль *FrmDbOptions* предоставляет доступ к опциям БД, а также позволяет изменить их. Данный модуль является вспомогательным для модуля *FrmSQLEx*, хотя может быть использован самостоятельно.
- 5.1.3) Модуль *FrmDbProcedures* служит для отображения хранимых на сервере процедур, функций и представлений БД. Является вспомогательным для модуля *FrmSQLEx* через модуль *WxDBHelp*.

- 5.1.4) Модуль *FrmDbTables* служит для отображения названий таблиц, принадлежащих БД. Кроме того, позволяет работать с этими таблицами, получая информацию об их содержимом, структуре, индексах, триггерах.
- 5.1.5) Модуль *FrmDbTriggers* служит для отображения триггеров таблиц, принадлежащих БД. Является вспомогательным для модуля *FrmDbTables* через модуль *WxDBHelp*.
- 5.1.6) Модуль *FrmSQLEx* при помощи функции *ExecuteSQLForm* вызывает анализатор SQL-запросов, подобный *Management Studio* для *Microsoft SQL Server*.
- 5.1.7) Модуль *WxDBHelpGUI* – реализация интерфейса для получения информации об объектах БД. Является вспомогательным для модулей: *FrmDBIndexes*, *FrmDBOptions*, *FrmDBTables*, *FrmDBTriggers*, *FrmDBProcedures*.
- 5.1.8) Модуль *WxDBSQLBackUpFrm* – реализация формы для резервного копирования данных.
- 5.1.9) Модуль *WxDBSQLPrmFrm* содержит реализацию формы *TFormSQLParams* для отображения SQL-параметров заданного SQL-запроса.
- 5.1.10) Модуль *WxObjectInspector* является вспомогательным для модуля *FrmSQLEx*. Позволяет настраивать свойства сетки при помощи *Object Inspector* по нажатию клавиши *F11*.
- 5.1.11) Модуль *WxSQLScriptGUI* содержит реализацию класса *TWxSQLScriptGUI* для SQL-сценариев для использования в модуле *FrmSQLEx*.
- 5.2) Модули без *GUI*, предоставляющие функциональность SQL-сервера.
- 5.2.1) Модуль *WxDBHelp* – служит для создания вспомогательной информационной базы для базы данных выбранной СУБД. Является вспомогательным для модулей: *FrmSQLEx*, *FrmDbIndexes*, *FrmDbOptions*, *FrmDbTables*, *FrmDbTriggers*, *FrmDbProcedures*.
- 5.2.2) Модуль *WxDBHelpASA* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *SYBASE ASA*.
- 5.2.3) Модуль *WxDBHelpASE* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *SYBASE ASE*.

- 5.2.4) Модуль *WxDbHelpDB2* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *DB2*.
- 5.2.5) Модуль *WxDbHelpEDB* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *ElevateDB*.
- 5.2.6) Модуль *WxDbHelpFirebird* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *Firebird*.
- 5.2.7) Модуль *WxDbHelpIFX* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *Informix*.
- 5.2.8) Модуль *WxDbHelpInterbase* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *Interbase*.
- 5.2.9) Модуль *WxDbHelpIQ* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *Sybase IQ*.
- 5.2.10) Модуль *WxDbHelpLinter* – реализация вспомогательной информационной базы для базы данных под управлением российской СУБД *ЛИНТЕР* от компании РЕЛЭКС.
- 5.2.11) Модуль *WxDbHelpMSSQL* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *Microsoft SQL SERVER*.
- 5.2.12) Модуль *WxDbHelpMySQL* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *MySQL*.
- 5.2.13) Модуль *WxDbHelpORACLE* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *ORACLE*.
- 5.2.14) Модуль *WxDbHelpPG* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *PostgreSQL*.
- 5.2.15) Модуль *WxDbSQLBackUp* предоставляет функциональность по резервному копированию данных.
- 5.2.16) Модуль *WxIBAdmin* содержит реализацию функциональности по резервному копированию для СУБД *Interbase*, *Firebird* и «РЕД БАЗА ДАННЫХ». Является вспомогательным для модулей *WxDbSQLBackUp* и *WxDbSQLBackUpFrm*.

- 5.2.17) Модуль *WxSQLScript* содержит реализацию класса *TWxSQLScript* для SQL-сценариев для использования в модуле *FrmSQLEx* (через промежуточный класс *TWxSQLScriptGUI*, определённый в модуле *WxSQLScriptGUI*).
- 5.2.18) Модуль *WxSQLText* является вспомогательным для модуля *WxSQLScript*. Содержит реализацию используемого в нём класса *TWxSQLText*.
- 5.2.19) Модуль *WxTypes* содержит набор констант и типов для выбора СУБД, с которыми будет осуществляться работа.
- 6) GUI-модули, предназначенные для работы с пользователями (размещены в подпапке *gui* исходников библиотеки *WXL*).
- 6.1) Модуль *WxUserDlg* содержит форму для добавления нового пользователя *WXL*-приложения.
- 6.2) Модуль *WxUserFrm* является главным модулем для осуществления удаления, редактирования и добавления пользователей *WXL*-приложения и их групп.
- 6.3) Модуль *WxUserGroupDlg* содержит форму для добавления новой группы пользователей *WXL*-приложения.
- 6.4) Модуль *WxUserRegDlg* обеспечивает безопасность при доступе к *WXL*-приложениям.
- 6.5) Модуль *WxUserTbl* является вспомогательным для других модулей работы с пользователями.
- 7) Модули, реализующие другие возможности.
- 7.1) GUI-модули, реализующие другие возможности (размещены в подпапке *gui* исходников библиотеки *WXL*).
- 7.1.1) Модуль *WxAppUtils* содержит необходимые библиотеке *WXL* формы. Также в данном модуле содержатся функции для работы с экранными подсказками, функции вызова сообщений с ошибками, предупреждениями, информацией, запросами на подтверждение.
- 7.1.2) Модуль *WxBitMask* содержит класс, ориентированный на удобную работу с *CheckBox*.
- 7.1.3) Модуль *WxCalendar* предназначен для работы с календарём.
- 7.1.4) Модуль *WxDlgScale* содержит реализацию диалога *TWxDialogScale* для настройки масштабирования форм в *WXL*-приложениях.

- 7.1.5) Модуль *WxInpStr* содержит реализацию диалога для ввода строки (с поддержкой различных строковых типов данных).
- 7.1.6) Модуль *WxLogDbGrid* содержит реализацию лога на основе сетки *WxGrid*.
- 7.1.7) Модуль *WxLogSynEdit* содержит реализацию лога с подсветкой синтаксиса на основе *TWxSynEdit*.
- 7.1.8) Модуль *WxLogVST* содержит реализацию лога на основе *TVirtualStringTree*.
- 7.1.9) Модуль *WxMaskEdit* содержит реализацию вспомогательного класса *TWxLabeledMaskEdit*. Используется в модуле *WxEntry*.
- 7.1.10) Модуль *WxMenuSettings* содержит реализацию формы для работы с настройками меню.
- 7.1.11) Модуль *WxSynEdit* предоставляет доступ к возможностям библиотеки компонентов *SynEdit*.
- 7.2) Модули без *GUI*, реализующие другие возможности.
- 7.2.1) Модуль *EpicTimer* содержит реализацию компонента таймера от *Tom Lisjac*. Описание компонента содержится на сайте <https://wiki.freepascal.org/EpikTimer>. Скачать компонент можно с сайта <https://github.com/graemeg/epiktimer>.
- 7.2.2) Модуль *WxAppCfg* содержит функциональность по работе с расположением приложения на жёстком диске.
- 7.2.3) Модуль *WxBits* содержит реализацию класса *TWxBits*, предназначенного для работы с отдельными битами блока памяти. Поддерживается до 2^{34} битов.
- 7.2.4) Модуль *WxBlanks* предназначен для работы с условно пустыми значениями переменных.
- 7.2.5) Модуль *WxCRC* содержит функциональность по вычислению контрольных сумм *CRC8*, *CRC32*, *CRC64*.
- 7.2.6) Модуль *WxDateUtils* содержит специальную функцию по переводу даты в строку.
- 7.2.7) Модуль *WxDbUtils* содержит набор вспомогательных функций для работы со стандартными классами *TField* и *TFieldDefs*.
- 7.2.8) Модуль *WxEncoding* содержит функциональность по работе с кодировками.

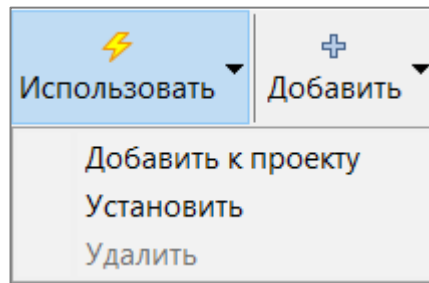
- 7.2.9) Модуль *WxKbd* содержит функциональность по переключению раскладок клавиатуры.
- 7.2.10) Модуль *WxLng* содержит языковые константы для возможности локализации WXL-приложений. Поддерживаются русский и английский языки.
- 7.2.11) Модуль *WxLogFile* содержит функциональность по работе с файлом журнала (лог-файлом).
- 7.2.12) Модуль *WiParamStr* является вспомогательным для модуля *WxAppCfg*. Содержит реализацию функций *HasOption* и *GetOptionValue* для проверки наличия опции и получения её значения.
- 7.2.13) Модуль *WxSort* содержит реализацию нескольких функций сортировки.
- 7.2.14) Модуль *WxStopWatch* содержит реализацию таймера *TWxStopWatch* для использования в других модулях WXL-библиотеки.
- 7.2.15) Модуль *WxStrSearch* содержит функциональность по поиску в строке и списке строк.
- 7.2.16) Модуль *WxStrUtils* содержит необходимые библиотеке WXL операции со строками.
- 7.2.17) Модуль *WxUnicode* позволяет работать с юникод-строками с учётом кодовых точек.
- 7.2.18) Модуль *WxVarUtils* содержит функциональность по работе с типом данных *variant*.

1.3 Установка библиотеки WXL

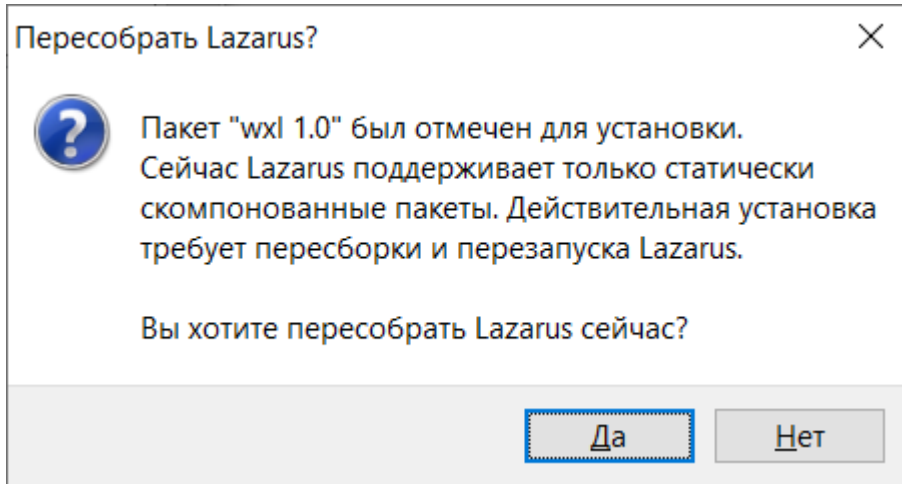
Рекомендуется установка компонентов и исходных кодов в «*d:\lazarus*».

Автоматические сценарии установки находятся в папках «*\wxl\lpk_build_windows*» (например, файл «*_build_lazarus32_laz-2.0.6-fpc-3.0.4.bat*») и «*\wxl\lpk_build_linux*» (например, файл «*_build_lazarus32_laz-2.0.6-fpc-3.0.4.sh*»).

Для ручной установки необходимо использовать следующую последовательность действий. Выбрать пункт главного меню «Пакет», «Открыть файл пакета», найти папку «*wxl\lpk*», выбрать файл «*wxl.lpk*», «Использовать», «Установить».



Далее необходимо подтвердить пересборку *Lazarus*.



Далее необходимо убедиться, что появилась вкладка компонентов *wx1*. Список установленных компонентов показан на рисунке.



В конкретном проекте необходимо также добавить путь к исходникам библиотеки *wx1*. «Проект», «Инспектор проекта», «Параметры», «Параметры компилятора», «Пути», «Другие модули». В зависимости от выбранной технологии доступа к данным также необходимо добавить путь к подпапке исходников *sqlldb*, *unidac* или *zeos*. Если используется один из генераторов отчётности, то необходимо также добавить путь к подпапке исходников *rpt*.

1.4 Настройка файла *wxdefs.inc*

Внешний вид файла «*wxdefs.inc*»:

```
{ $DEFINE stMSSQL }
{ . $DEFINE stORACLE }
{ . $DEFINE stASE }
{ . $DEFINE stASA }
{ . $DEFINE stIQ }
{ $DEFINE stINTERBASE }
{ $DEFINE stFIREBIRD }
{ . $DEFINE stMYSQL }
{ . $DEFINE stLINTER }
{ . $DEFINE stDB2 }
{ . $DEFINE stEDB }
```

```

{$DEFINE stPOSTGRESQL}
{.$DEFINE stINFORMIX}

{.$DEFINE USESYNEDIT}

{.$DEFINE USETDBF}

{.$DEFINE USESQLDB}
{.$DEFINE USESQLDB_WXL}

{.$DEFINE USEUNIDAC}

{$DEFINE USEZEOS}
{$DEFINE USEZEOS_WXL}

{$DEFINE USEBUFDATASET}
{.$DEFINE USEVIRTUALTABLE}
{.$DEFINE USEKBMEMTABLE}

{.$DEFINE USEISAMUSERTABLES}

{.$DEFINE USEFASTREPORT}
{.$DEFINE USELAZREPORT}

{$DEFINE DISABLEPRINTERS}

```

Файл *wxdefs.inc* содержит набор директив условной компиляции и должен использоваться во всех WXL-приложениях. В типовом WXL-приложении он обычно используется в модулях «ПРИЛОЖЕНИЕconnectdata» («*travelsconnectdata*»), «ПРИЛОЖЕНИЕdataman» («*travelsdataman*») и «ПРИЛОЖЕНИЕdatamod» («*travelsdatamod*»). Например, его подключение может выглядеть следующим образом:

```

unit TravelsDataMan;

{$MODE OBJFPC}
{$H+}

{$I wxdefs.inc}

```

В зависимости от потребностей Вашего приложения, Вы должны модифицировать файл *wxdefs.inc* таким образом, чтобы использовалось нужное Вам сочетание директив. Для каждого проекта, с которым Вы работаете в данный момент, модифицируйте *wxdefs.inc* под конкретные нужды проекта. Чтобы случайно не забыть установить необходимые директивы, можно использовать следующий прием:

```

unit TravelsDataMan;

{$MODE OBJFPC}
{$H+}

{$I wxdefs.inc}
{Этот код не позволит скомпилировать проект при неправильно назначенных директивах}

```

```
{IFDEFDEF USETDBF}
  USETDBF -- если не выбрано "USETDBF", при компиляции здесь получим ошибку
{ENDIF}
```

Ниже приводится описание директив условной компиляции, сосредоточенных в «*wxdefs.inc*»:

```
{Выбор используемых СУБД.}
{$DEFINE stMSSQL}
{.$DEFINE stORACLE}
{.$DEFINE stASE}
{.$DEFINE stASA}
{.$DEFINE stIQ}
{$DEFINE stINTERBASE}
{$DEFINE stFIREBIRD}
{.$DEFINE stMYSQL}
{.$DEFINE stLINTER}
{.$DEFINE stDB2}
{.$DEFINE stEDB}
{$DEFINE stPOSTGRESQL}
{.$DEFINE stINFORMIX}
```

Если есть точка, то указанная СУБД **не** поддерживается. Если нет точки, то указанная СУБД **используется**.

Затрагиваемые модули:

frmdbindexes, frmdbprocedures, frmdbtables, frmsqlx, wxconnectstring, wxdbhelp, wxdbhelpgui, wxdbhelpasa.pas, wxdbhelpase.pas, wxdbhelpdb2.pas, wxdbhelpedb.pas, wxdbhelpfirebird.pas, wxdbhelpifx.pas, wxdbhelpinterbase.pas, wxdbhelpiq.pas, wxdbhelplinter.pas, wxdbhelpmssql.pas, wxdbhelpmysql.pas, wxdbhelporacle.pas, wxdbhelppg.pas, wxdbsql.pas, wxdbsqlbackup.pas, wxdbsqlbackupfrm, wxdbsql_fpc.pas, wxdbsql_uni.pas, wxdbsql_zeos.pas, wxsqltext, wxsynedit.pas, wxtypes.pas.

```
{Использовать ли SYNEDIT для подсветки текстов?}
{$DEFINE USESYNEDIT}
```

Если есть точка, то *SYNEDIT* **не** используется. Если нет точки, то *SYNEDIT* **используется**.

Затрагиваемые модули:

wxdataset.pas, wxdbhelpgui.pas, wxlogsynedit.pas, wxsynedit.pas, wxuserfrm.pas, wxuserregdlg.pas, wxusertbl.pas.

```
{Использовать ли TDBF?}
{$DEFINE USETDBF}
```

Если есть точка, то *TDBF* **не** используется. Если нет точки, то *TDBF* **используется**.

Затрагиваемые модули:

wxdataset.pas, *wxdatasetexport.pas*, *wxdatasetexportfrm.pas*, *wxdatasetinfo.pas*,
wxdatasetview.pas, *wxdbhelpgui.pas*, *wxdbisam.pas*, *wxlogsynedit.pas*, *wxsynedit.pas*,
wxuserfrm.pas, *wxuserregdlg.pas*, *wxusertbl.pas*.

```
{Выбор технологии доступа к базам данных SQL-сервера.}  
{.$DEFINE USESQLDB}  
{.$DEFINE USESQLDB_WXL}  
  
{.$DEFINE USEUNIDAC}  
  
{.$DEFINE USEZEOS}  
{.$DEFINE USEZEOS_WXL}
```

Если есть точка, то указанная технология **не** используется. Если нет точки, то указанная технология **используется**.

Следует выбрать **только одну** из технологий *SQLDB*, *UNIDAC*, *ZEOS*. Использование библиотеки *ZeosDBO* является предпочтительным. Выбор *USESQLDB_WXL* оказывает влияние только в случае выбора *USESQLDB*. Выбор *USEZEOS_WXL* оказывает влияние только в случае выбора *USEZEOS*.

Затрагиваемые модули:

wxconnectstring.pas, *wxdataset.pas*, *wxdatasetsort.pas*, *wxdbhelp.pas*,
wxdbhelpfirebird.pas, *wxdbhelpgui.pas*, *wxdbsql.inc*, *wxdbsql.pas*,
wxdbsqlbackup.pas, *wxdbsqlconnectdlg.pas*, *wxdbsqlconnectlist.pas*,
wxdbsqlconnectpanel.pas, *wxdbsqlproxy.pas*, *wxibadmin.pas*, *wxtypes.pas*,
wxuserfrm.pas, *wxusertbl.pas*.

```
{Какой набор данных в памяти использовать}  
{.$DEFINE USEBUFDATASET}  
{.$DEFINE USEVIRTUALTABLE}  
{.$DEFINE USEKBMMEMTABLE}
```

Если есть точка, то указанный набор данных в памяти **не** используется. Если нет точки, то указанный набор данных в памяти **используется**. Следует выбрать **только один** из наборов данных в памяти *BUFDATASET*, *VIRTUALTABLE*, *KBMMEMTABLE*. Использование набора данных в памяти *BUFDATASET* является предпочтительным.

Затрагиваемые модули:

wxdataset.pas, *wxdatasetexport.pas*, *wxdatasetexportfrm.pas*, *wxdatasetinfo.pas*,
wxdatasetsort.pas, *wxdatasetstoragebin.pas*, *wxdatasetstoragejson.pas*,

wxdatasetstoragetext.pas, wxdatasetstoragexml.pas, wxdatasetview.pas, wxlogdbgrid.pas.

```
{Выбор способа хранения таблицы групп пользователей  
WXL-приложения и таблицы пользователей WXL-приложения.}  
{.$DEFINE USEISAMUSERTABLES}
```

Если есть точка, то указанная технология (локальные таблицы) **не** используется. Если нет точки, то указанная технология **используется**.

Затрагиваемые модули:

wxuserfrm.pas, wxuserregdlg.pas, wxusertbl.pas.

```
{Какой генератор отчётности использовать}  
{.$DEFINE USEFASTREPORT}  
{.$DEFINE USELAZREPORT}
```

Если есть точка, то указанный генератор отчётности **не** используется. Если нет точки, то указанный генератор отчётности **используется**. Если в приложении требуется использование генератора отчётности, то следует выбрать **только один** из генераторов отчётности *FASTREPORT, LAZREPORT*. Генерация отчётности подробно освещена в главе 8.

Затрагиваемые модули:

wxtblrpt.pas, wxtblrptexp.pas.

```
{Отключать ли печать при просмотре текстовых файлов?}  
{$DEFINE DISABLEPRINTERS}
```

Если есть точка, то печать при просмотре текстовых файлов (модуль *wtxtview*) **возможна**. Если нет точки, то печать при просмотре текстовых файлов (модуль *wtxtview*) **отключена**.

Затрагиваемые модули:

wtxtview.pas.

2. Используемые компоненты в составе *Lazarus*

2.1 *SQLdb*

Назначение *SQLdb*

SQLdb – это свободная (лицензии *GPL*, *LGPL*) и открытая библиотека компонентов для доступа к данным, которая поддерживается на различных операционных системах и разрабатывается сообществом разработчиков *Free Pascal* + *Lazarus*. Технология *SQLdb* используется в среде программирования *Lazarus* для доступа к базам данных по умолчанию, то есть она входит в «коробочный» вариант поставки среды *Lazarus*. Начало разработки *SQLdb* датируется 2004 годом. В *SQLdb* поддерживается работа с *Microsoft SQL Server*, *Sybase ASE*, *PostgreSQL*, *ORACLE*, *MySQL*, *SQLite*, *Interbase* и *Firebird*. Кроме того, для работы с иными СУБД возможно использование технологии *ODBC* через базовый компонент *TSQLConnector* и соответствующие драйверы *ODBC*.

Для работоспособности взаимодействия с выбранной СУБД необходимо наличие соответствующих клиентских библиотек. Используемую клиентскую библиотеку можно указать либо в наследниках компонента соединения *TSQLConnection*, либо при помощи специализированного компонента *TSQLDbLibraryLoader*.

Приобретение *SQLdb*

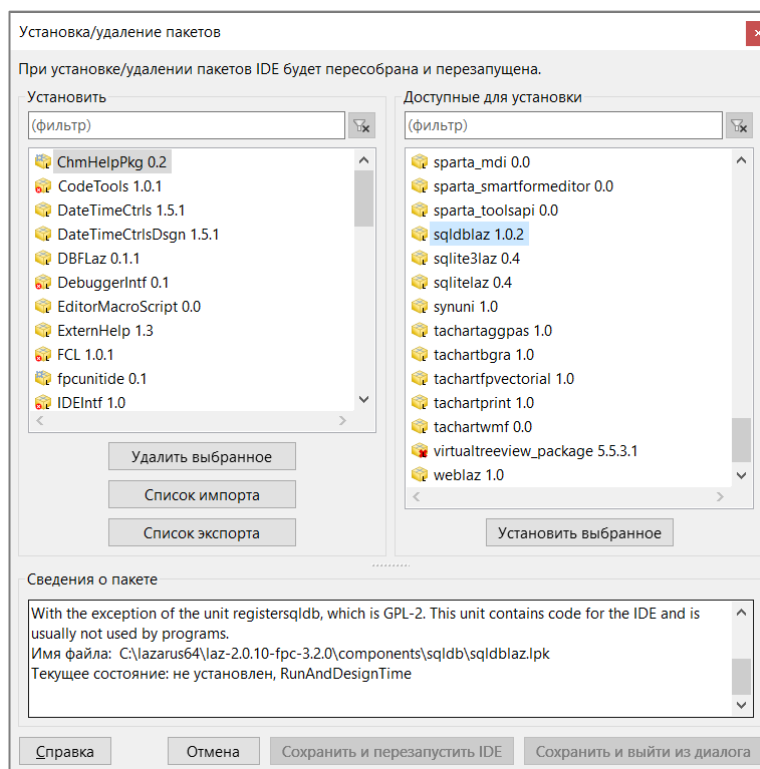
SQLdb является бесплатным программным обеспечением, входит в стандартную поставку *Lazarus*.

Установка *SQLdb*

Вкладка *SQLDb* уже может присутствовать в *Lazarus*. В этом случае никаких дополнительных действий по установке предпринимать не нужно.

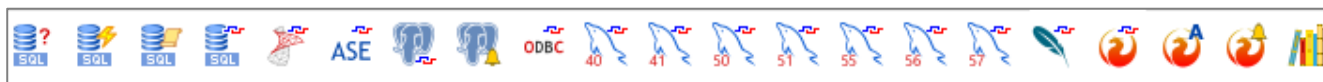
Если вкладка отсутствует, то рекомендуется следующий порядок действий. Выбрать пункт главного меню «Пакет», «Установить/Удалить пакеты ...», выбрать *SQLDBLaz* присутствующей в списке версии (например, *SQLDBLaz 1.0.2*). Нажать кнопку «Установить выбранное», нажать кнопку «Сохранить и перезапустить *IDE*».

Пример формы для выбора компонентов для установки показан на рисунке:



Далее необходимо пересобрать *Lazarus*.

После этого необходимо убедиться, что появилась вкладка компонентов «*SQLdb*». Список установленных компонентов показан на рисунке.



Использование *SQLdb* в библиотеке *WXL*

Библиотека *WXL* может использовать *SqlDB* для работы с различными СУБД, если в конкретном проекте в файле «*wxdefs.inc*» установлена директива условной компиляции `{ $DEFINE USESQLDB }`.

Использование *SQLdb* в проектах

Для использования *SQLdb* в проектах не требуется предпринимать никаких дополнительных действий.

2.2 *SynEdit*

Назначение *SynEdit*

SynEdit представляет собой набор компонентов с открытым исходным кодом для *Delphi* и *Lazarus* для подсветки текстов в соответствии с заданным синтаксисом. В *Lazarus* используется для подсветки синтаксиса внутри самой *IDE*.

Текущая версия для *Lazarus* (на момент написания документации, июнь 2022) поддерживается *Martin Friebe*.

Приобретение *SynEdit*

SynEdit является бесплатным программным обеспечением, входит в стандартную поставку *Lazarus*. Официальная страница набора компонентов находится по адресу: <http://synedit.sourceforge.net>.

Установка *SynEdit*

Компоненты *SynEdit* не требуется устанавливать, вкладка *SynEdit* присутствует в *Lazarus* по умолчанию. Список установленных компонентов показан на рисунке.

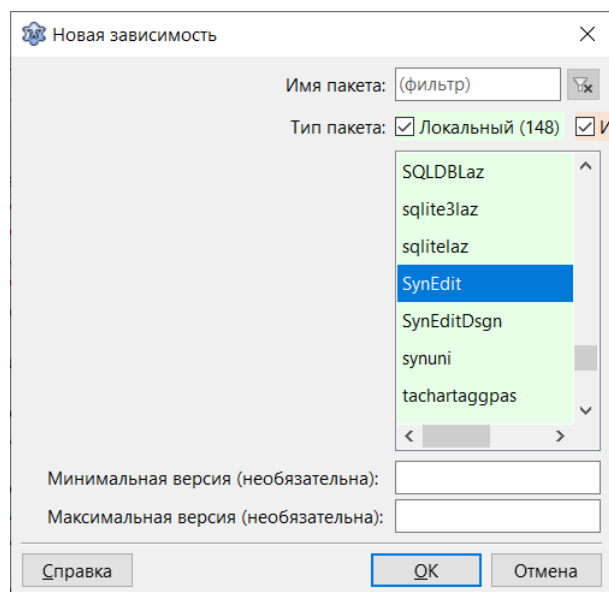


Использование *SynEdit* в библиотеке *WXL*

Библиотека *WXL* может использовать *SYNEDIT* для подсветки текстов, если в конкретном проекте в файле «*wxdefs.inc*» установлена директива условной компиляции `{$DEFINE USESYNEDIT}`. В библиотеке *WXL* *SynEdit* используется посредством модуля *WxSynEdit* и предназначается для подсветок текстов синтаксиса языка *Pascal*, а также для подсветки *SQL*-запросов в стиле поддерживаемых СУБД. Причём подсветка в стиле синтаксиса языка *Pascal* производится, главным образом, для показа подсвеченного текста на отдельной форме, в то время как подсветки для *SQL*-запросов используются, главным образом, в анализаторе запросов модуля *FrmSQLEx*.

Использование *SynEdit* в проектах

В самом проекте необходимо добавить пакет *SynEdit* к «Требуемые пакеты». Для этого в главном меню *Lazarus* необходимо выбрать «Проект», «Инспектор проекта», затем нажать правой кнопкой мыши на «Требуемые пакеты» на появившейся форме, выбрать «Добавить». Появится следующая форма:



Он будет добавлен автоматически при размещении компонентов из набора на форме.

2.3 LazReport

Назначение *LazReport*

LazReport представляет собой включаемый в поставку *Lazarus* (хотя и не предустановленный) набор компонентов для генерации отчётов. В его основе лежит генератор отчётов *FreeReport*. В основе *FreeReport* лежит *FastReport* версии 2.3 (выпуск 2002 года). Приведём таблицу сравнения возможностей *FreeReport* и *FastReport* версии 6.

Характеристики	FreeReport 2.3	FastReport
Скриптовый язык	Да	Да
Подсветка синтаксиса в редакторе скрипта	Да	Да
Смена band datasource из скрипта	Нет	Да
Доступ из скрипта к объектам на других страницах отчёта	Нет	Да
Функция FORMATTEXT, оператор FOR, процедуры EXIT, INC и DEC в FastReport Pascal	Нет	Да
Поддержка Interbase Express (IBX), IObjects, ActiveX Data Objects (ADO)	Да	support more dbs
Delphi6 dbExpress DBXComponents	Нет	Да
Инспектор объектов	Упрощённый	Улучшенный
Cross-tab отчеты	Базовый	Улучшенный
Диалоги	Нет	Да
Line style	Да	Да
BarCode	Да	Да
RTF 2.0	Да	Да
Компонент TfrPreview для создания собственного предварительного просмотра	Да	Да
Компонент TfrPrintTable	Нет	Да

Опции для объекта "Text": толщина линий и символов, отступ сверху и слева, "Скрывать повторяющиеся значения"	Нет	Да
Запрет редактирования, перемещения, изменения размера и удаления объектов	Нет	Да
Свойство бендов: PrintChildIfInvisible	Нет	Да
Свойство бенда GroupHeader: Master	Нет	Да
Словарь данных	Нет	Да
Свойство BandAlign для всех объектов отчета	Нет	Да
Свойство объекта Text: HideZeros (скрывать нулевые)	Нет	Да
TfrReport.OnObjectClick event	Нет	Да
Свойство TfrReport.MDIPreview	Нет	Да
Свойство TfrReport.PrintIfEmpty TfrDesigner.OpenDir, SaveDir	Нет	Да
Свойство TfrPage.Visible	Нет	Да
Фильтры экспорта	txt, htm, csv, rtf	txt, htm, csv, rtf, xls, etc
Языки	8	21
Локализация инспектора	Нет	Да
Поддержка нескольких языков одновременно	Нет	Да
Поддержка языков "справа-налево"	Нет	Да

В большинстве задач для создания простых отчётов в кроссплатформенных приложениях *Lazarus* достаточно использования *LazReport*. В том случае, если его функциональности оказывается недостаточно, есть прямой резон купить и использовать *FastReport*.

Приобретение *LazReport*

LazReport является бесплатным программным обеспечением, входит в стандартную поставку *Lazarus*.

Установка *LazReport*

Компоненты *LazReport* не установлены по умолчанию в *Lazarus*.

Рекомендуется следующий порядок действий. Выбрать пункт главного меню «Пакет», «Установить/Удалить пакеты ...», выбрать *LazReport* присутствующей в списке версии (например, *LazReport* 0.9.9). Нажать кнопку «Установить выбранное». Далее необходимо нажать кнопку «Сохранить и перезапустить IDE» и подтвердить пересборку *Lazarus*. При необходимости экспорта отчётов в формат *PDF* необходимо проделать аналогичные действия для пакета *lr_PDFExport* присутствующей в списке версии (например, *lr_PDFExport* 0.9).

Следует отметить, что возможно ограничиться только одной пересборкой *Lazarus*, если сразу выбрать и тот, и другой пакеты.

Список установленных компонентов на вкладке *LazReport*:



Список установленных компонентов на вкладке *LazReport* при установке пакета *lr_PDFExport*:



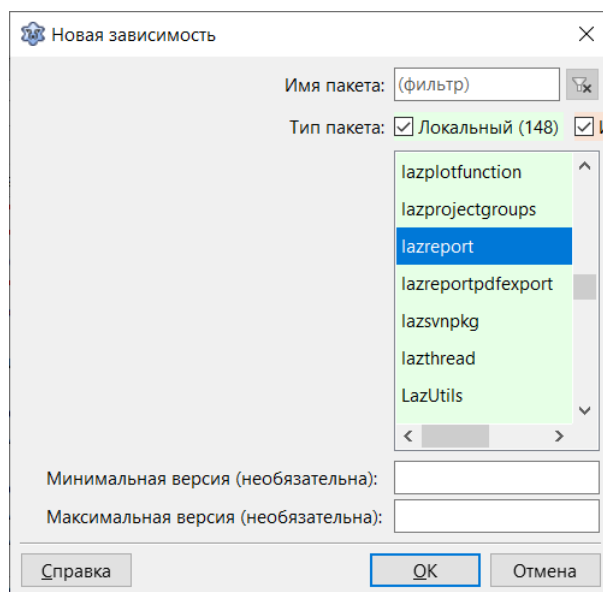
Добавлен один компонент *TlrPDFExport* .

Использование *LazReport* в библиотеке *WXL*

Библиотека *WXL* может использовать *LazReport* для генерации отчётности, если в конкретном проекте в файле «*wxdefs.inc*» установлена директива условной компиляции `{$DEFINE USELAZREPORT}`. В библиотеке *WXL* *LazReport* используется только в модуле *WxTblRpt* для построения отчётности.

Использование *LazReport* в проектах

В самом проекте необходимо добавить *LazReport* к «Требуемые пакеты». Для этого в главном меню *Lazarus* необходимо выбрать «Проект», «Инспектор проекта», затем нажать правой кнопкой мыши на «Требуемые пакеты» на появившейся форме, выбрать «Добавить». Появится следующая форма:



Он будет добавлен автоматически при размещении компонентов из набора на форме.

2.4 TDBF

Назначение TDBF


TDBF представляет собой бесплатный компонент прямого доступа для *Delphi* и *Lazarus*. Созданные при помощи данного средства программы для работы с базами данных компактны и не требуют предварительной инсталляции.

Подходит для работы с таблицами *DBASE*, *Clipper* и *Visual FoxPro*.

Приобретение TDBF

TDBF является бесплатным программным обеспечением, входит в стандартную поставку *Lazarus*. Официальная страница компонента находится по адресу: <http://tdbf.sourceforge.net/>.

Установка TDBF

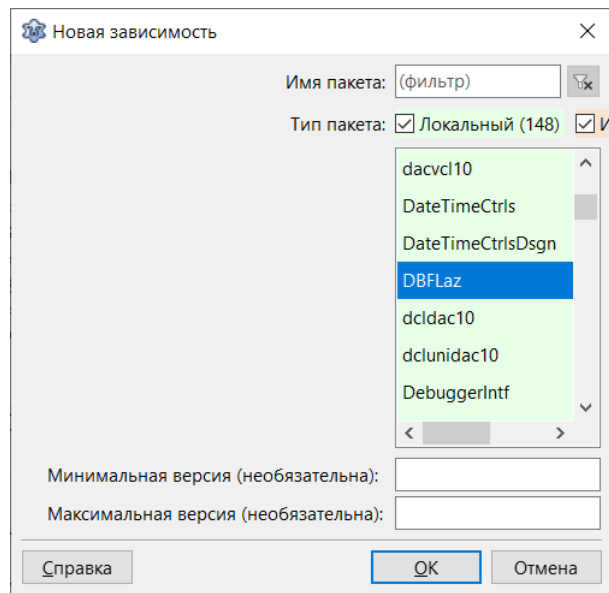
Компонент *TDBF* не требуется устанавливать, он находится на вкладке *Lazarus Data Access* по умолчанию. Внешний вид компонента: .

Использование TDBF в библиотеке WXL

Библиотека *WXL* может использовать *TDBF* для доступа к локальным таблицам, если в конкретном проекте в файле «*wxdefs.inc*» установлена директива условной компиляции `{ $DEFINE USETDBF }`. Единый подход к использованию всех выбираемых методов доступа к локальным базам данных реализован в модуле *WxDBISAM*.

Использование TDBF в проектах

В самом проекте (если не было добавлено автоматически, например, при размещении компонента *TDBF* на форме) необходимо добавить *DbfLaz* к «Требуемые пакеты». Для этого в главном меню *Lazarus* необходимо выбрать «Проект», «Инспектор проекта», затем нажать правой кнопкой мыши на «Требуемые пакеты» на появившейся форме, выбрать «Добавить». Появится следующая форма:



Пример отображения таблицы при помощи *TDBF*

- 1) Создаём новый проект («Файл», «Создать», «Приложение», «ОК»).
- 2) Помещаем на форму две панели (с *Align = alTop* и *Align = alClient*), компонент класса *TDBF*, источник данных класса *TDataSource*. На нижнюю панель кладём сетку класса *TDbGrid* (с *Align = alClient*), а на верхнюю – кнопку.
- 3) Устанавливаем набор данных *DataSet* источника данных на экземпляр класса *TDBF* (например, *Dbf1*), источник данных *DataSource* сетки на экземпляр класса *TDataSource* (например, *DataSource1*).
- 4) По нажатию кнопки устанавливаем свойство *TableName* компонента *Dbf1* на полный адрес таблицы *DBASE*, *FoxPro* или *Clipper*; свойство *TableLevel Dbf1* на требуемый уровень совместимости (например, 3 для совместимости с *DBASE3+* или 25 для совместимости с *FoxPro*).
- 5) Устанавливаем свойство *Active* в *true* и видим требуемую таблицу на сетке.

3. Связные библиотеки

3.1 ZeosDBO

Назначение ZeosDBO

ZeosDBO – это свободная (лицензии *GPL*, *LGPL*) и открытая библиотека компонентов для доступа к данным, которая поддерживается на различных операционных системах (*Windows*, *Linux* и *FreeBSD*) и разрабатывается сообществом разработчиков *ZeosLib*. Начало разработки *ZeosLib* датируется 1999 годом. В настоящее время поддерживаются следующие источники данных: *MySQL*, *PostgreSQL*, *Interbase*, *Firebird*, *Microsoft SQL Server*, *Sybase ASE*, *ORACLE*, *SQLite*. Компоненты доступа к данным на основе библиотеки *ZeosDBO* не являются встроенными ни в среду программирования *Delphi*, ни в среду программирования *Lazarus*, но их возможно скачать и установить в обеих средах. Начиная с версии 7.3.x, в библиотеке *ZeosLib* реализована поддержка *ADO*, *OLE DB* и, что особенно важно, – *ODBC*.

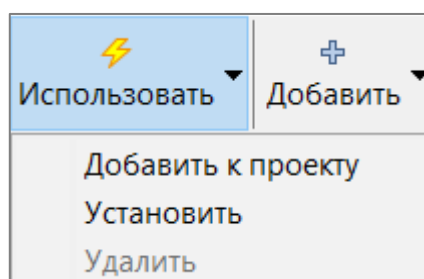
Приобретение ZeosDBO

ZeosDBO является бесплатным программным обеспечением. Официальная страница компонентов находится по адресу: <https://zeoslib.sourceforge.io/>. Скачать компоненты можно по адресу <https://sourceforge.net/projects/zeoslib/>. Для выбора последней версии необходимо выбрать «*Subversion*», «*Download Snapshot*».

Установка ZeosDBO

Рекомендуется установка компонентов и исходных кодов в «*d:\lazarus*».

Для ручной установки необходимо использовать следующую последовательность действий. Выбрать пункт главного меню «Пакет», «Открыть файл пакета», найти папку «*\zeos\packages\lazarus*», выбрать файл «*zcomponentdesign.lpk*», «Использовать», «Установить».



Далее необходимо подтвердить пересборку *Lazarus*.

После этого необходимо убедиться, что появилась вкладка компонентов «*Zeos Access*». Список установленных компонентов показан на рисунке.

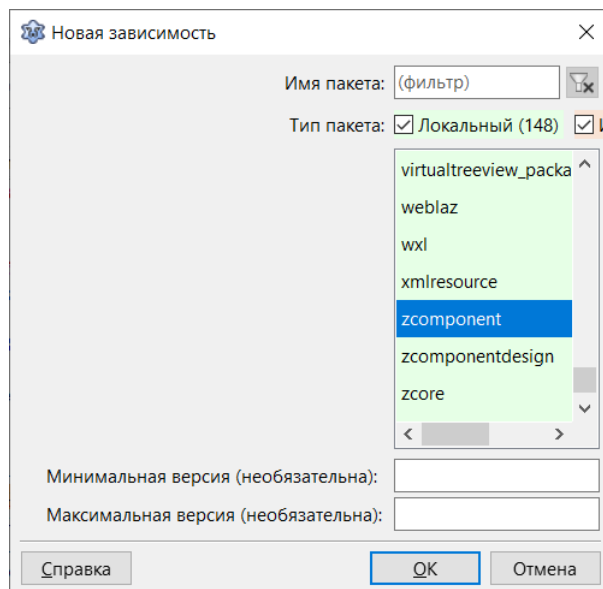


Использование *ZeosDBO* в библиотеке *WXL*

Библиотека *WXL* может использовать *ZeosDBO* для связи с базами данных под управлением различных СУБД, если в конкретном проекте в файле «*wxdefs.inc*» установлена директива условной компиляции `{ $DEFINE USEZEOS }`. Эта директива является директивой по умолчанию.

Использование *ZeosDBO* в проектах

В самом проекте (если не было добавлено автоматически, например, при размещении компонентов на форме) необходимо добавить *ZComponent* к «Требуемые пакеты». Для этого в главном меню *Lazarus* необходимо выбрать «Проект», «Инспектор проекта», затем нажать правой кнопкой мыши на «Требуемые пакеты» на появившейся форме, выбрать «Добавить». Появится следующая форма:



На ней необходимо будет выбрать «*zcomponent*» и нажать «ОК». Для исключения «*zcomponent*» из списка зависимостей можно выбрать «Проект», «Инспектор проекта». Затем нажать правой кнопкой мыши на «*zcomponent*» и выбрать «Удалить».

3.2 UniDAC

Назначение UNIDAC

UniDAC (Universal Data Access Components) – это универсальная проприетарная технология доступа к данным от компании *Devart*, которая поддерживается на различных операционных системах (*Windows, Mac OS X, iOS, Android, Linux* и *FreeBSD*) для 32-битных и 64-битных приложений. Компания *Devart* была основана в 1997 году и с самого своего основания специализировалась на технологиях доступа к данным. К настоящему моменту её продуктами пользуются более 40,000 пользователей, а сама компания является партнёром *Microsoft (Silver Application Development Partner)* и *ORACLE (Silver Partner in the Oracle Partner Network (OPN))*. Первая версия продукта *UniDAC* вышла 23 апреля 2008 года. То есть сама компания существует уже более 20 лет, а её продукт *UniDAC* более 10 лет. Компоненты доступа к данным на основе технологии *UniDAC* не являются встроенными ни в среду программирования *Delphi*, ни в среду программирования *Lazarus*, но их возможно купить и установить в обеих средах (в том числе, *с исходным кодом*). При использовании технологии *UniDAC* возможно работать с различными источниками данных напрямую, без использования клиентских библиотек (*Microsoft SQL Server, ORACLE, MySQL, PostgreSQL, SQLite, NexusDB*), через использование *DB Client* и/или *ODBC*. В качестве независимых компонентов компания *Devart* разработала компоненты для доступа к отдельным СУБД: *ODAC (Oracle Data Access Components)*, *SDAC (SQL Server Data Access Components)*, *MyDAC (Data Access Components for MySQL)*, *IBDAC (InterBase Data Access Components)*, *PgDAC (PostgreSQL Data Access Components)*, *LiteDAC (SQLite Data Access Components)*.

Приобретение UNIDAC

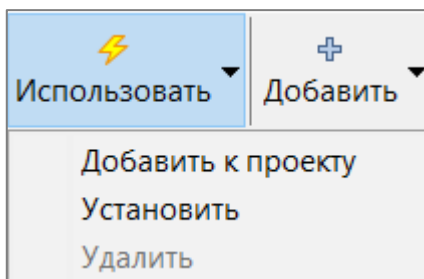
Компоненты *UNIDAC* можно приобрести по адресу <https://softonline.com.ua/catalog/devart/>.

Установка UNIDAC

Рекомендуется установка компонентов и исходных кодов в «*d:\lazarus*».

Автоматические сценарии установки находятся в подпапке «*wxl*» (например, файл «*_build_lazarus32_laz-2.0.6-fpc-3.0.4.bat*»).

Для ручной установки необходимо использовать следующую последовательность действий. Выбрать пункт главного меню «Пакет», «Открыть файл пакета», найти папку «*%UniDAC%\Source\Lazarus1*», выбрать файл «*dclunidac10.lpk*», «Использовать», «Установить».



Далее необходимо подтвердить пересборку *Lazarus*.

После этого необходимо убедиться, что появилась вкладка компонентов «*UniDAC*». Список установленных компонентов показан на рисунке:



Кроме того, аналогичным образом требуется установить провайдеры для выбранных СУБД.

Провайдер	СУБД
<i>tdsprovider10.lpk</i>	<i>Microsoft SQL Server, Sybase ASE, Sybase ASA</i>
<i>odbcprovider10.lpk</i>	Любые СУБД через технологию доступа к данным <i>ODBC</i>
<i>msprovider10.lpk</i>	<i>Microsoft SQL Server</i>
<i>oraprovider10.lpk</i>	<i>Oracle</i>
<i>aseprovider10.lpk</i>	<i>Sybase ASE</i>
<i>ibprovider10.lpk</i>	<i>Interbase, Firebird, Ред База Данных</i>
<i>myprovider10.lpk</i>	<i>MySQL, MariaDB</i>
<i>db2provider10.lpk</i>	<i>DB2</i>
<i>pgprovider10.lpk</i>	<i>PostgreSQL, Postgres Pro</i>
<i>liteprovider10.lpk</i>	<i>SQLite</i>

Компоненты для провайдеров устанавливаются на вкладку «*UniDAC Providers*». Список установленных компонентов показан на рисунке:



Использование *UNIDAC* в библиотеке *WXL*

Библиотека *WXL* может использовать *UNIDAC* для связи с базами данных под управлением различных СУБД, если в конкретном проекте в файле «*wxdefs.inc*»

установлена директива условной компиляции `{$DEFINE USEUNIDAC}`. Эта директива не является рекомендуемой.

Использование *UNIDAC* в проектах

В самом проекте (если не было добавлено автоматически, например, при размещении компонентов на форме) необходимо добавить пакет *unidac10* и пакеты требуемых провайдеров к «Требуемые пакеты». Для этого в главном меню *Lazarus* необходимо выбрать «Проект», «Инспектор проекта», затем нажать правой кнопкой мыши на «Требуемые пакеты» на появившейся форме, выбрать «Добавить».

3.3 *FastReport*

Назначение *FastReport*

FastReport представляет собой набор компонентов для построения отчётов от компании *Fast Reports, Inc.* Он представляет собой сочетание дизайнера, генератора и *Preview* отчётов. Он позволяет экспортировать отчёт в форматы *BMP, JPG, TIF, PNG, TXT, CSV, RTF, ODT, ODS, HTML, DOCX, XLSX, PPTX, PDF, XML, ZPL, PS, PPML, DBF*. В полученном отчёте возможно выполнять поиск текста.

Fast Reports, Inc – компания по разработке программного обеспечения для формирования отчётов. Основана в 1998 году. Российский офис и служба технической поддержки находятся в городе Ростов-на-Дону. Партнёры и дилеры компании *Fast Reports* работают во многих странах мира.

История продуктов *FastReport*

- 1998–2002 гг. – создание компании и продвижение флагманского продукта, генератора отчётов *FastReport VCL*. Попутно поддерживаются две платформы: *Win32* и *Kylix*. В середине 2002 года *FastReport VCL* был признан лучшим генератором отчётов года, по результатам опроса *delphizine.com*.
- В мае 2003 года выходит новая версия *FastReport 2.5 VCL* со множеством исправлений и дополнений, среди которых:
 - добавлен экспорт в черно-белом варианте *BMP, JPEG* и *TIFF*;
 - экспорт в *RTF*, но на тот момент в бета-тесте;

- добавлены новые свойства и возможности, связанные с функциональностью дизайнера отчётов – улучшенный *CrossTab*;
 - возможность отключения подсветки синтаксиса в редакторе скрипта;
 - программное изменение страниц перед печатью;
 - функция *FieldIsNull* в скрипте.
- В сентябре 2003 года релиз библиотеки *FastScript*. *FastScript* включает в себя 4 скриптовых языка: *Pascal Script*, *C++Script*, *JScript*, *BasicScript*. *FastScript* даёт возможность встроить скрипты, написанные на любом из четырёх языков, в отчёт, чем расширяет возможности *FastReport*.
 - 30 августа 2004 года состоялся релиз *FastReport 3 VCL* – следующей версии продукта для *Delphi*-разработчиков. Релиз данного продукта толкнул разработчиков в сторону клиент-серверной архитектуры. После чего появилась возможность создания полноценного сервера отчётов. Основная особенность «тройки» – полная многопоточность, что позволило встраивать в многозадачные среды (в том числе, клиент-серверные, для *web*-отчетности).
Возможности клиент-серверной архитектуры:
 - построение отчётов любой сложности на стороне сервера по запросу клиента без непосредственного доступа клиента к серверу баз данных;
 - обслуживание нескольких клиентов сервером в различных потоках позволяет добиться высокой нагрузочной способности и минимизации времени;
 - применение протокола передачи данных *HTTP (RFC 2068)* позволяет использовать большое количество существующих программ, таких как *web*-браузеры (*Internet Explorer*, *Netscape Navigator*, *Mozilla*, *Opera* и др.), *Proxy*-серверы, *web*-серверы (*Internet Information Server*, *Apache* и др.) для совместной работы без дополнительных трудоёмких решений;
 - применение технологий сжатия на основе алгоритма *GZip (RFC 1952)* уменьшает сетевой трафик и увеличивает общую производительность клиент-сервер системы;
 - использование в качестве клиента не только внутреннего компонента *FastReport*, но и любого *web*-браузера.

- 22 августа 2005г. на основе дизайнера отчётов *FastReport* был выпущен продукт, ориентированный на конечных пользователей, *FastReport Studio*. *FR Studio* явил собой полноценный дизайнер отчётов для пользователей, не владеющих знаниями языков программирования, но знающих, что такое база данных, и умеющих с ней работать. Начиная с 2013 года, продукт больше не поддерживается. Причина – устаревшая технология.
- 30 мая 2006 года состоялся релиз *FastReport Server*. В марте 2008 года выходит вторая версия *FastReport Server* с изменённым ядром на *FastReport 4 VCL*, а также множеством дополнительных исправлений и баг-фиксов. Начиная с 2013 года, продукт не поддерживается. Причина – устаревшая технология.
- Сентябрь-октябрь 2006 года – бета-тестирование и выход новой, четвёртой, версии генератора отчётов *FastReport VCL*. Интервал между версиями составил 2 года. В «четвёрку» были включены все возможности *FastReport 3 VCL*, а также большое количество нововведений. Большая часть изменений коснулась ядра продукта и улучшения дизайнера отчётов.
- Примерно через год, в середине августа 2007 года в связи с высокими требованиями пользователей к продуктам *FastReport*, компания выпускает набор *Delphi*-компонентов для *OLAP*-анализа *FastCube*. Продукт содержит библиотеку *FastScript* и обладает возможностью интеграции с *FastReport 4 VCL*. Другими словами, у пользователей появилась возможность отправить полученный анализ *FastCube* в отчётный документ *FastReport*'а, либо запускать его в окне дизайнера отчётов.
- 16 марта 2009 года состоялся официальный релиз *FastReport.Net* для *Microsoft Visual Studio*, написан полностью на *C#*, поддерживает *ASP.NET* и *MVC*.
- 2012 год: выходит продукт для новой среды разработки, предлагаемой компанией *Embarcadero – FireMonkey*. *FastReport FMX* работает как в среде *Windows*, так и в *MacOS*. Кроме того, *Fast Reports* выпускает кроссплатформенный генератор отчётов *FastReport.Mono*. По сути, это адаптированный *FastReport.Net*.

- 2013 год – выход новых версий *FastReport.Net* с индексом 2013, *FastCube 2 VCL* для новой *RAD Studio XE5*. В октябре был запущен сервер демо-отчётов, построенный на технологии *MVC*.
- 1 апреля 2014 года – выход новой версии генератора отчётов *FastReport VCL 5*. Добавлены новые классы, объекты (*2D* штрих-коды, *Code128*, *EAN128* с автоматической кодировкой), улучшена интерактивность, новые экспорты в *HTML5 (div)*, *DOCX*, *XLSX*, *PPTX*, улучшен экспорт в *PDF*, изменения в *GUI*, также улучшения клиент-серверной части.
- 2016 год – выход *FastReport for Lazarus beta*.
- 2018 год – выход новой версии генератора отчётов *FastReport VCL 6* с поддержкой *Lazarus*.

Таким образом, с 2018 года имеется возможность использования *FastReport 6* при разработке кроссплатформенных приложений на *Lazarus*.

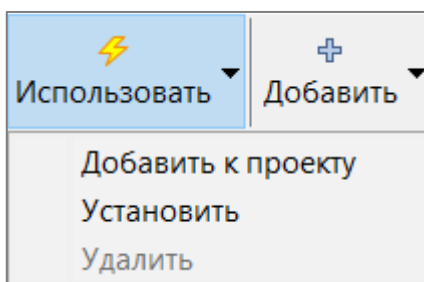
Приобретение *FastReport*

Пробные версии *FastReport* можно скачать бесплатно, но полную функциональность обеспечивает только полная версия. Купить полную версию, а также скачать пробную можно по адресу <http://www.fast-report.com/>.

Установка *FastReport*

Рекомендуется установка компонентов и исходных кодов в «*d:\lazarus*».

Для ручной установки необходимо использовать следующую последовательность действий. Выбрать пункт главного меню «Пакет», «Открыть файл пакета», найти папку «*%FastReport%\fastscript*», выбрать файл «*fs_lazarus.lpk*», «Использовать», «Установить».



Далее необходимо подтвердить пересборку *Lazarus*.

После этого аналогично необходимо установить пакет «*fr6_lazarus.lpk*» из папки «*%FastReport%\Source*». Если требуется экспорт данных, то необходимо установить пакет «*frxeb_lazarus.lpk*» из папки «*%FastReport%\Source\ExportPack*». Если требуется функциональность по построению диаграмм, то необходимо установить пакет «*frxchartlazarus*» из папки «*%FastReport%\Source\lazchart*».

После этого необходимо убедиться, что появились требуемые вкладки компонентов:

- Список установленных компонентов вкладки «*FastScript*»:



- Список установленных компонентов вкладки «*FastReport 6.x*»:

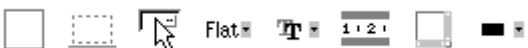


- Список установленных компонентов вкладки «*FastReport 6.x*» при установке пакета *lazchart*:



Единственное отличие – наличие компонента *TfrxChartObject*  (самый правый в списке).

- Список установленных компонентов вкладки «*FR6 Tools*»:



- Список установленных компонентов вкладки «*FastReport 6.0 Exports*»:



Использование *FastReport* в библиотеке *WxLib*

Библиотека *WXL* может использовать *FastReport* для генерации отчётности, если в конкретном проекте в файле «*wxdefs.inc*» установлена директива условной компиляции `{$DEFINE USEFASTREPORT}`. В библиотеке *WXL* *FastReport* используется только в модуле *WxTblRpt* для построения отчётности.

4. WXL-утилиты

4.1 Утилита *wxlgen*

4.1.1 Что первично в программировании приложений баз данных

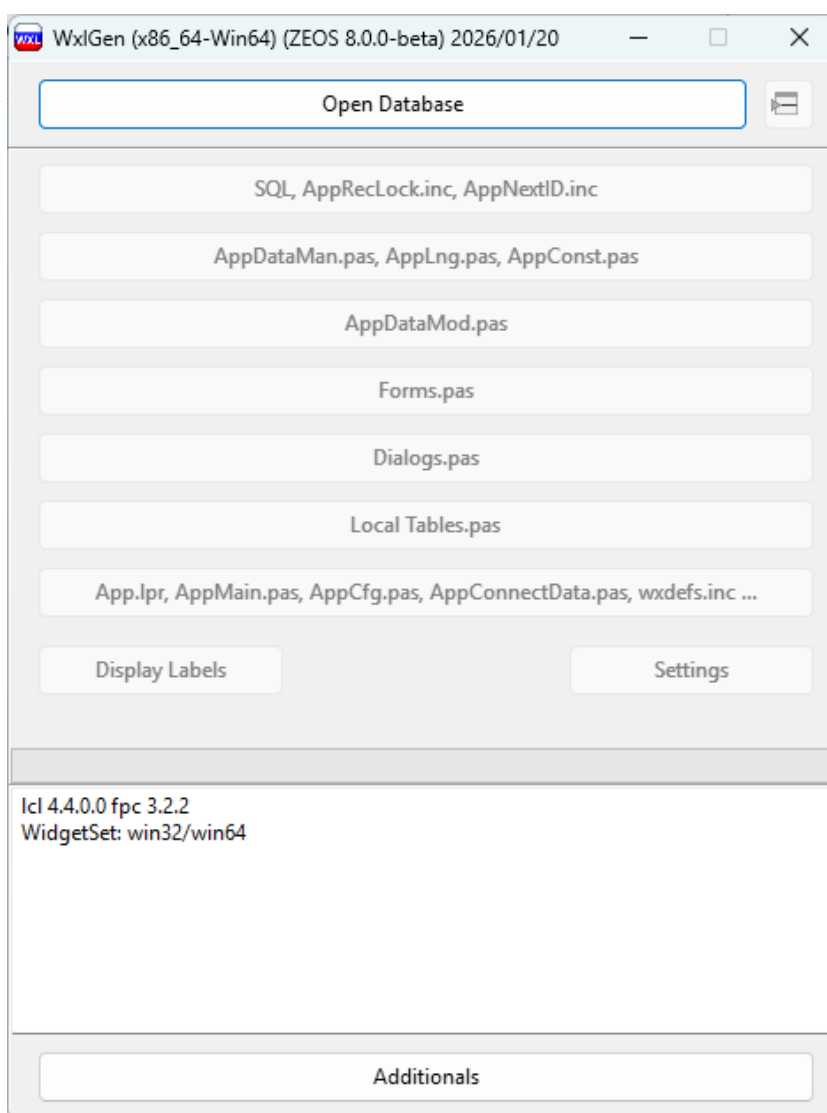
При создании приложений, работающих с базами данных, часто возникает вопрос, что первично: создание базы данных, таблиц и хранимых процедур или создание приложения, работающего с этой базой данных. Хорошо спланированная база данных играет роль крепкого фундамента для приложения, позволяет значительно облегчить автоматическую генерацию исходного кода и ресурсов приложения. Для проектирования реляционных баз данных наиболее часто применяются метод нормализации таблиц и метод *ER*-диаграмм. Генератор *wxlgen*, поставляемый вместе с библиотекой *WXL*, позволяет автоматически генерировать *SQL*-скрипты для объектов баз данных, которые затем используются в любом типом *WXL*-приложении по работе с базами данных. Таким образом, на первом месте стоит продумывание структуры базы данных и создание необходимых объектов для неё. Такой подход позволяет осуществлять основную функциональность приложения по работе с базами данных путём вызова соответствующих хранимых на сервере процедур. Кроме того, нет необходимости снова и снова менять структуру базы данных, чтобы подладить её под конкретное приложение, а затем проводить сложную работу по согласованию нескольких приложений, использующих одну и ту же базу данных. Таким образом, мы пришли к выводу о предпочтительности такого подхода к созданию любого *WXL*-приложения:

- 1) Создание базы данных и её таблиц при помощи одного из подходов: метода нормализации таблиц или *ER*-метода.
- 2) Добавление в таблицы необходимых индексов, триггеров, ограничений.
- 3) Автоматическая генерация наиболее часто встречаемых хранимых на сервере процедур для базы данных при помощи генератора *wxlgen*.
- 4) Добавление необходимых для решения конкретной задачи хранимых на сервере процедур.

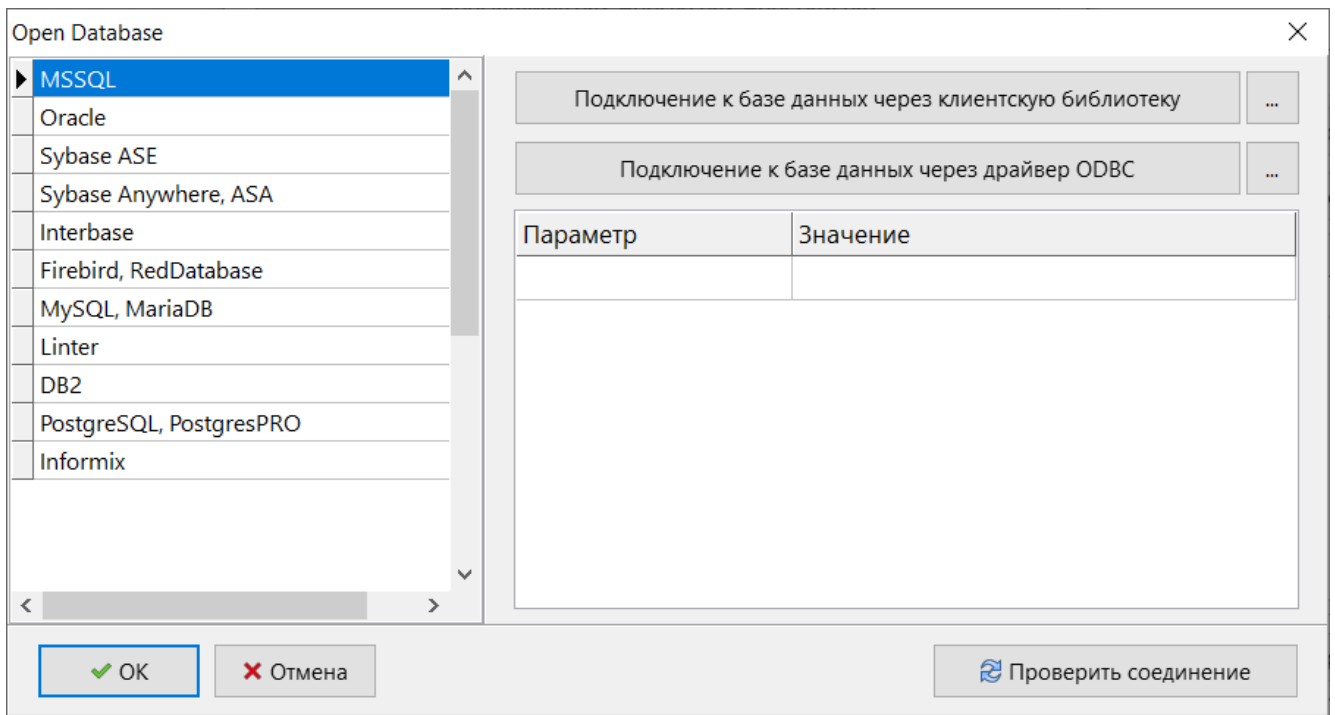
- 5) Создание исходных текстов и ресурсов для приложения по работе с данной базой данных при помощи генератора *wxlgen*.
- 6) Внесение необходимых изменений в зависимости от поставленных задач.

4.1.2 Описание генератора *wxlgen*


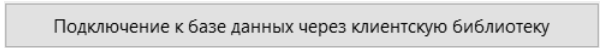
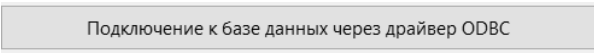
- 1) Запуск генератора *wxlgen* осуществляется запуском одного из поставляемых вместе с библиотекой *WXL* исполняемых файлов: *wxlgen_zeos.exe* (используется технология *ZeosDBO*) или *wxlgen_sqldb* (используется технология *SQLdb*).
- 2) В результате появляется форма с двумя доступными кнопками «*Open Database*» и «*Additional*s».

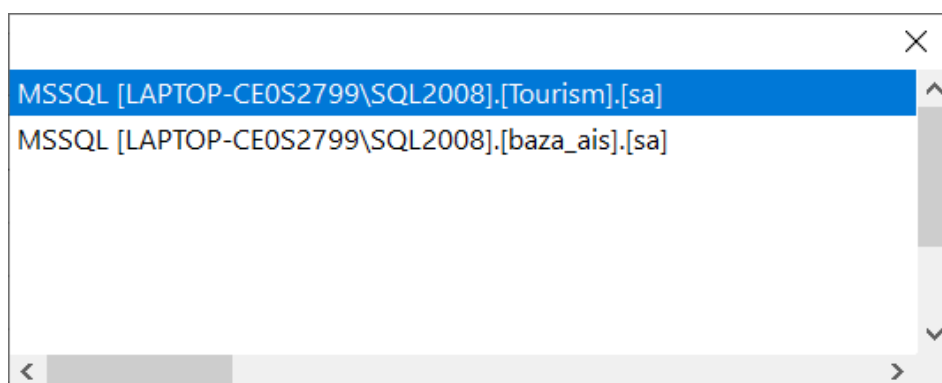


В первую очередь, необходимо выбрать *SQL*-сервер и базу данных, нажав «*Open Database*». В результате появляется форма выбора *SQL*-сервера и базы данных:

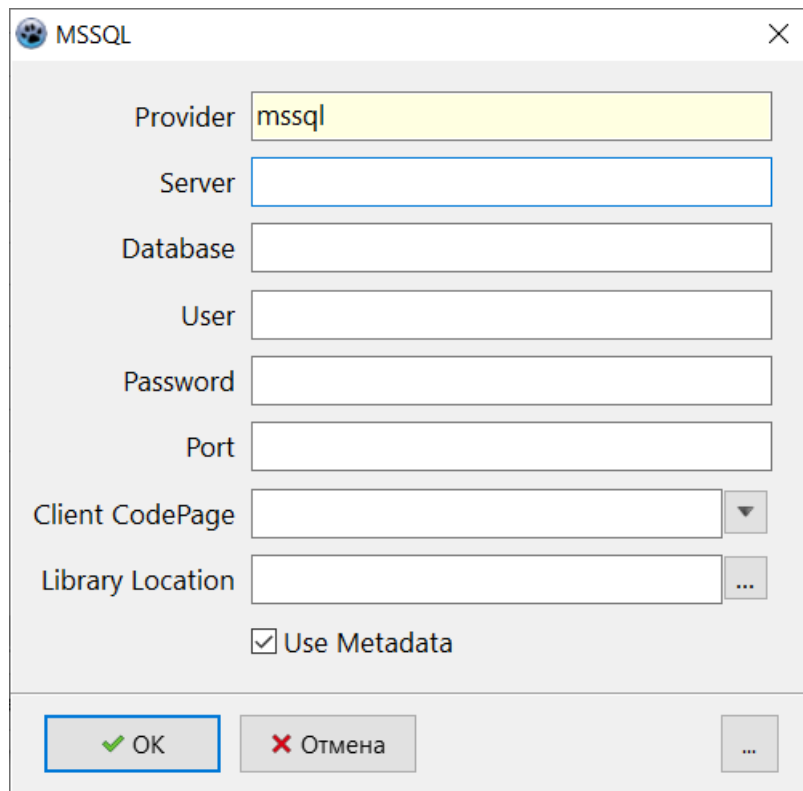



При помощи сетки в левой части формы можно выбрать тип *SQL*-сервера: «*MSSQL*», «*Oracle*», «*Sybase ASE*», «*Sybase Anywhere, ASA*», «*Interbase*», «*Firebird, RedDatabase*», «*MySQL, MariaDB*», «*Linter*», «*DB2*», «*PostgreSQL, PostgresPRO*», «*Informix*». Доступ через *ODBC* при помощи кнопки «Подключение к базе данных через драйвер *ODBC*» доступен для любой СУБД, доступ через клиентскую библиотеку при помощи кнопки «Подключение к базе данных через клиентскую библиотеку» возможен не для всех СУБД. Далее необходимо выбрать способ соединения с базой данных на сервере, нажав соответствующую кнопку.

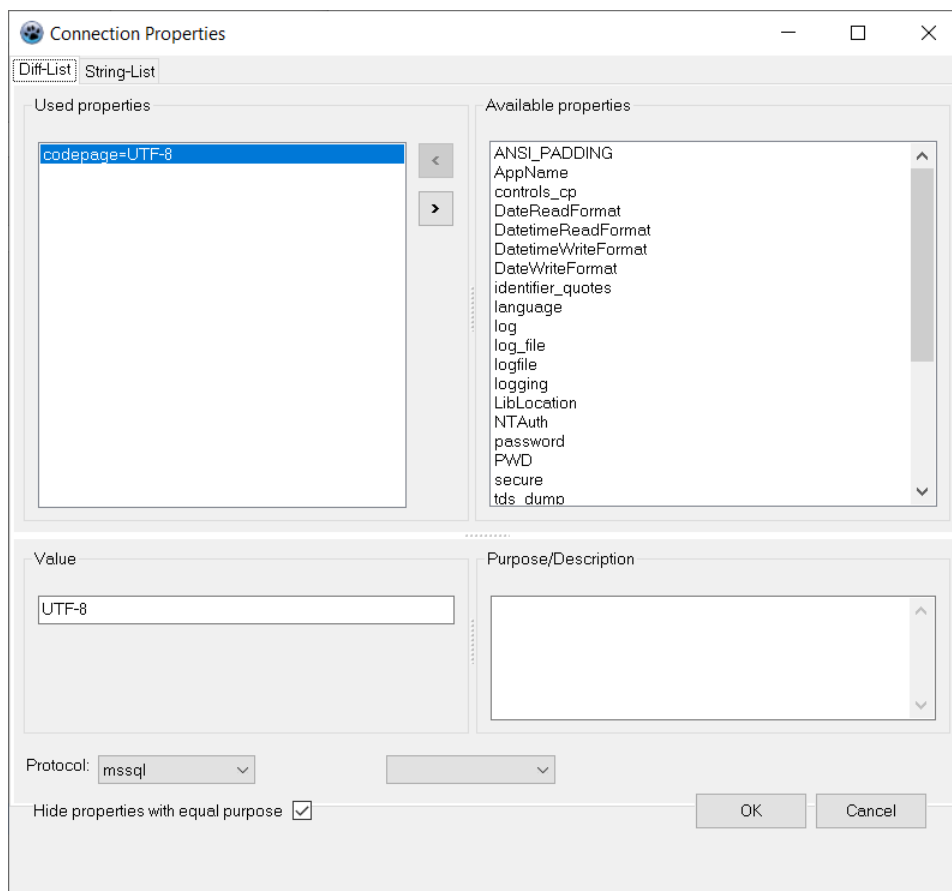
Использование кнопок  справа от кнопок  и  позволяет выбрать одно из используемых ранее подключений. Внешний вид формы для выбора показан на рисунке:



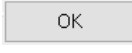
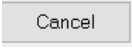


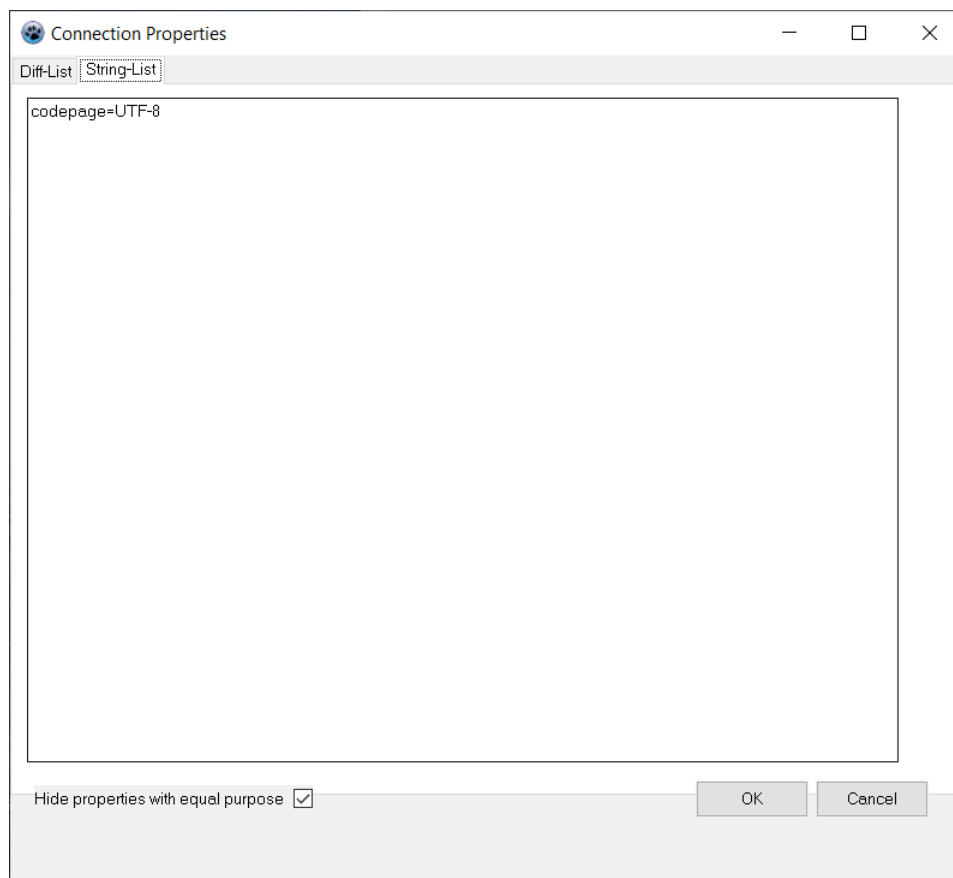
Внешний вид диалога соединения при выборе «Подключение к базе данных через клиентскую библиотеку» на примере *Microsoft SQL Server*:


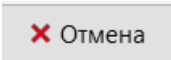


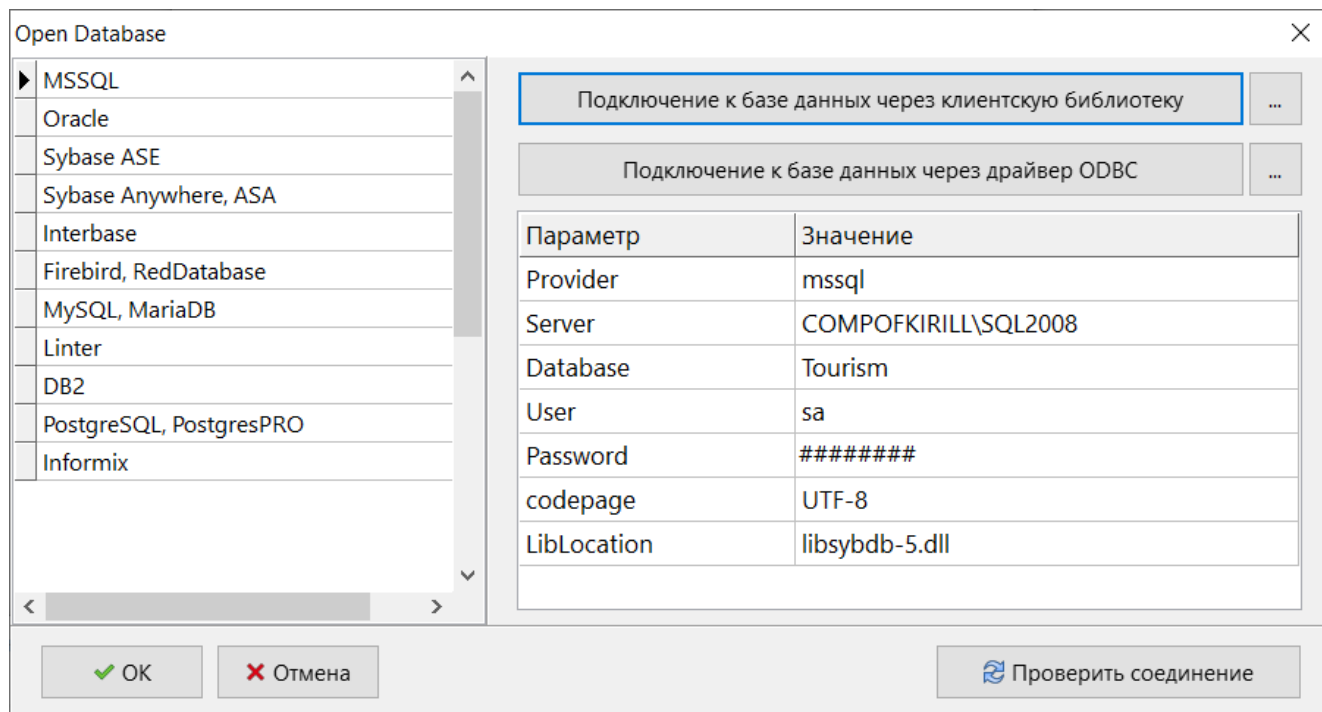
Кнопка  служит для отображения дополнительных параметров соединения, специфичных для выбранной СУБД. Внешний вид формы их выбора показан на рисунке:

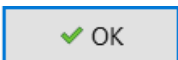
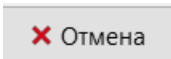
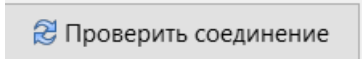


Для выбора дополнительного параметра необходимо выбрать его значение в списке «*Available properties*». В этом случае назначение выбранного параметра будет отображено в поле ввода под меткой «*Purpose/Description*», метка «*Value*» будет заменена на название параметра. После этого необходимо заполнить поле ввода под названием параметра его значением и нажать на кнопку . Для исключения выбора дополнительного параметра необходимо выбрать его значение в списке «*Used properties*» и нажать на кнопку . Не рекомендуется изменять значение из списков «*Protocol*», так как это нарушит согласованность между выбранной СУБД и предлагаемым набором дополнительных параметров. Флажок «*Hide properties with equal purpose*» позволяет исключить параметры, которые имеют разное наименование, но одинаковое назначение. Кнопкой  можно подтвердить введённые дополнительные параметры соединения, кнопкой  на диалоге соединения можно отказаться от них. Изменив вкладку «*Diff-List*» на вкладку «*String-List*» можно задать те же значения в формате «*Name=Value*».

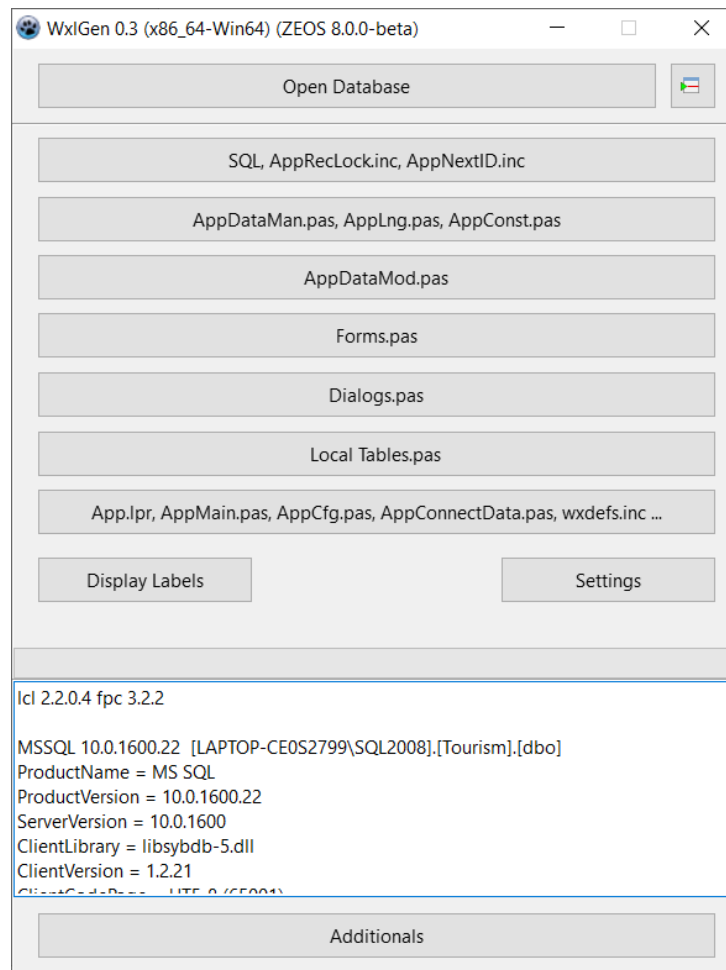


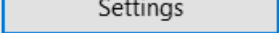
Кнопкой  на диалоге соединения можно подтвердить введённые параметры соединения, кнопкой  на диалоге соединения можно отказаться от них. Внешний вид формы после подтверждения введённых параметров:

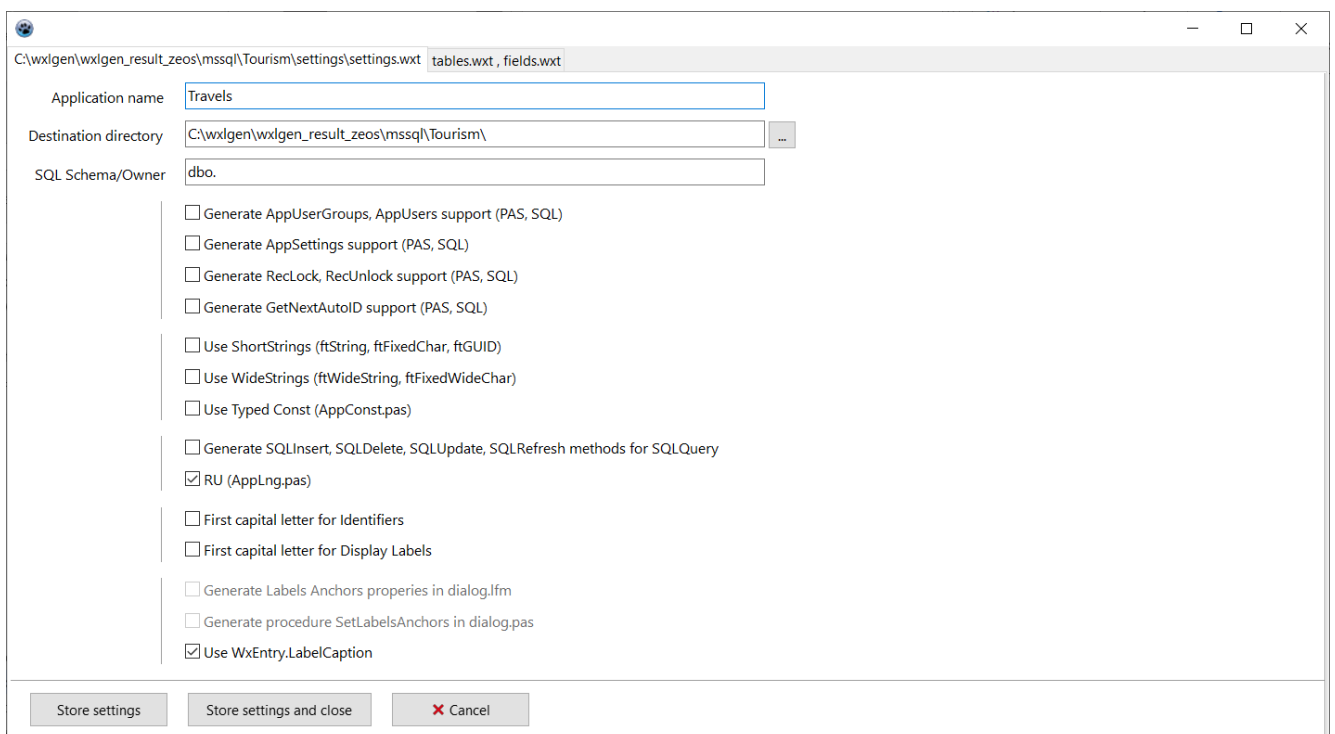



Кнопкой  можно подтвердить введённые параметры соединения, кнопкой  можно отказаться от них, кнопка  служит для проверки соединения.

3) После заполнения параметров на диалоге и их подтверждения активными становятся все кнопки главной формы, а в нижней части формы появляется информация о версии *SQL*-сервера, его типе и имени, информация об имени выбранной базы данных и используемой клиентской библиотеки. Директория для сохранения по умолчанию в общем случае имеет вид «Директория *wxlggen.exe*\wxlggen_result_zeos\СУБД\БД» (например, она может выглядеть следующим образом: «C:\wxlggen\wxlggen_result_zeos\MSSQL\Tourism»), она создаётся сразу же после подтверждения параметров. Общий вид главной формы показан на рисунке:



В первую очередь, необходимо нажать кнопку  для выбора соответствующих опций при генерации исходных текстов и ресурсов приложения и необходимых *SQL*-скриптов. При её нажатии появляется следующая форма:



Компонент ввода справа от метки «*Application name*» позволяет выбрать имя приложения (влияет на имена создаваемых файлов), компонент ввода справа от метки «*Destination directory*» позволяет выбрать директорию для сохранения файлов (кнопка  позволяет выбрать её при помощи стандартного диалога), компонент ввода справа от метки «*SQL Schema/Owner*» позволяет выбрать название схемы или владельца в зависимости от выбранной ранее СУБД. Рассмотрим функциональность первой (показанной на рисунке) вкладки «ПУТЬ*settings.wxt*» (для рассматриваемого примера имеет вид «*C:\wxmlgen\wxmlgen_result_zeos\mssql\Tourism\settings\settings.wxt*»).

3.1) Опция «*Generate AppUserGroups, AppUsers support (PAS, SQL)*» служит для выбора, включать ли генерацию *SQL*-скриптов для создания таблиц пользователей и их групп для контролируемого доступа к *WXL*-приложению. Кроме того, данная опция включает функциональность по работе с этими таблицами в генерируемый код приложения. Данная функциональность подробно рассмотрена в главе 11.

3.2) Опция «*Generate AppSettings support (PAS, SQL)*» служит для выбора, включать ли генерацию *SQL*-скриптов по созданию таблицы пользовательских настроек *WxlAppSettings*. Кроме того, данная опция определяет, включать ли функциональность по работе с этой таблицей в генерируемый код приложения.

3.3) Опция «*Generate RecLock, RecUnlock support (PAS, SQL)*» служит для выбора, включать ли функциональность по работе со многими пользователями в исходные тексты приложения и создаваемые *SQL*-скрипты. Данная функциональность подробно рассмотрена в разделе 10.6.

3.4) Опция «*Generate GetNextAutoID support (PAS, SQL)*» служит для выбора, включать ли функциональность по генерации уникальных идентификаторов в исходные тексты приложения и создаваемые *SQL*-скрипты. Данная функциональность подробно рассмотрена в разделе 10.5.

3.5) Опция «*Use ShortStrings (ftString, ftFixedChar, ftGUID)*» указывает, использовать ли для строковых переменных в записях, соответствующих

таблицам, тип *shortstring* вместо типа *string*. При наличии опции в качестве компонентов ввода-вывода используются компоненты *TWxEntryShortString*.

3.6) Опция «*Use WideStrings (ftWideString, ftFixedWideChar)*» указывает, использовать ли для строковых переменных в записях, соответствующих таблицам, тип *widestring* вместо типа *string*. При наличии опции в качестве компонентов ввода-вывода используются компоненты *TWxEntryWideString*.

3.7) Опция «*Use Typed Const (AppConst.pas)*». Регистр идентификаторов в модуле «ПРИЛОЖЕНИЕConst.pas» («*travelsconst.pas*») определяется в зависимости от используемой СУБД через функцию *IdentifierCase*. В модуле «ПРИЛОЖЕНИЕConst.inc» («*travelsconst.inc*») константы имеют тип (например, «*fnId: string = 'Id';*» вместо «*fnId = 'Id';*»)

3.8) Опция «*Generate SQLInsert, SQLDelete, SQLUpdate, SQLRefresh methods for SQLQuery*» служит для выбора, создавать ли процедуры по выполнению вставки, редактирования, удаления и обновления записей (*SetInsertClause, SetUpdateClause, SetDeleteClause, SetRefreshClause*) в модуле «ПРИЛОЖЕНИЕdataman.pas» («*travelsdataman.pas*»).

3.9) Опция «*RU (AppLng.pas)*». Если опция используется, то файл «ПРИЛОЖЕНИЕlng.pas» («*travelslng.pas*») формируется на русском языке. Если не используется, то файл «ПРИЛОЖЕНИЕlng.pas» («*travelslng.pas*») формируется на английском языке.

3.10) Опция «*First capital letter for Identifiers*». Данная опция определяет, требуется ли заменять первую букву в идентификаторах на заглавную, если в самой базе данных она была строчной.

3.11) Опция «*First capital letter for Display Labels*». Данная опция определяет, требуется ли заменять первую букву в дескрипторах на заглавную, если в самой базе данных она была строчной.

3.12) Опция «*Generate Labels Anchors properties in dialog.lfm*». Данная опция возможна только при отсутствии опций «*Use WxEntry.LabelCaption*» и «*Generate procedure SetLabelsAnchors in dialog.pas*». На диалогах (модули «DLGИМЯТАБЛИЦЫ.PAS» (например, «*dlghotels.pas*»)) для компонентов

ввода-вывода создаются соответствующие метки (*TLabel*), процедура для регулирования их расположения не создаётся.

3.13) Опция «*Generate procedure SetLabelsAnchors in dialog.pas*». Данная опция возможна только при отсутствии опций «*Use WxEntry.LabelCaption*» и «*Generate Labels Anchors properties in dialog.lfm*». На диалогах (модули «*DLГИМЯТАБЛИЦЫ.PAS*» (например, «*dlghotels.pas*»)) для компонентов ввода-вывода создаются соответствующие метки (*TLabel*), их расположение регулируется процедурой *SetLabelsAnchors*.

3.14) Опция «*Use WxEntry.LabelCaption*». На диалогах (модули «*DLГИМЯТАБЛИЦЫ.PAS*» (например, «*dlghotels.pas*»)) на компонентах ввода-вывода используется свойство *LabelCaption*, отвечающее за присоединённую метку. Дополнительные компоненты-метки (*TLabel*) не создаются.

Рассмотрим функциональность второй вкладки – «*tables.wxt, fields.wxt*».

Следует иметь в виду, что для использования функциональности данной вкладки следует предварительно нажать кнопку «*AppDataMan.pas, AppLng.pas, AppConst.pas*» и выбрать те таблицы, с которыми будет осуществляться работа. Именно данные о выбранных таблицах и их полях отобразятся на форме.

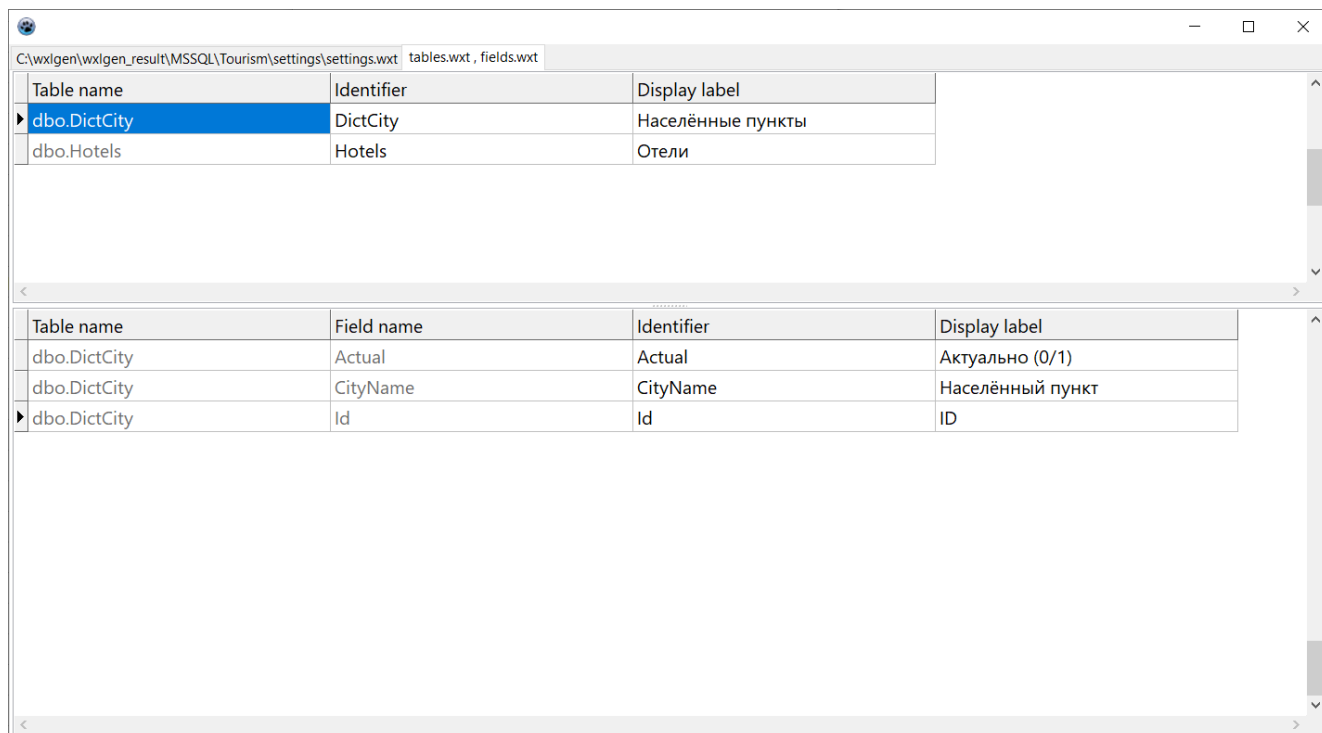


Table name	Identifier	Display label
dbo.DictCity	DictCity	Населённые пункты
dbo.Hotels	Hotels	Отели

Table name	Field name	Identifier	Display label
dbo.DictCity	Actual	Actual	Актуально (0/1)
dbo.DictCity	CityName	CityName	Населённый пункт
dbo.DictCity	Id	Id	ID

Идентификатор таблицы (*Identifier*) – это имя таблицы в программе. Таким образом, модули, создаваемые для работы с таблицей, будут иметь наименования «*FrmИдентификаторТаблицы*» (например, «*frmhotels.pas*» и «*frmhotels.lfm*»), «*DlgИдентификаторТаблицы*» (например, «*dlghotels.pas*» и «*dlghotels.lfm*»). Диалог для добавления и редактирования записей будет иметь наименование «*TDialogИдентификаторТаблицы*» («*TDialogHotels*»). Форма для работы с таблицей будет называться «*TFormИдентификаторТаблицы*» («*TFormHotels*»). Запись с данными таблицы в модуле «ПРИЛОЖЕНИЕ*dataman.pas*» («*travelsdataman.pas*») будет называться «*TDataИдентификаторТаблицы*» («*TDataHotels*»), а указатель на эту запись «*PDataИдентификаторТаблицы*» («*PDataHotels*»). Типизированный набор данных, предназначенный для работы с этой таблицей, также определённый в модуле «ПРИЛОЖЕНИЕ*dataman.pas*» («*travelsdataman.pas*»), будет иметь наименование «*TDataSetИдентификаторТаблицы*» («*TDataSetHotels*»).

Дескриптор таблицы (*Display label*) заносится в константу «*tdИдентификаторТаблицы*» («*tdHotels*») (префикс *td* является сокращением от *table descriptor*) модуля «ПРИЛОЖЕНИЕ*lng.pas*» («*travelslng.pas*»). Его использование необходимо добавить самостоятельно при необходимости отобразить русское наименование таблицы.

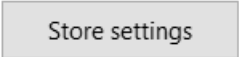
Идентификаторы полей таблицы (*Identifier*) используются для наименований компонентов в диалоге, предназначенном для добавления/редактирования записей таблицы. Например, если следовать рисунку, компонент ввода-вывода, отвечающий полю «*CityName*» таблицы «*dbo.DictCity*» будет называться «*WxEntryCityName*», а метка для него «*LabelCityName*». Кроме того, в модуле «ПРИЛОЖЕНИЕ*dataman.pas*» («*travelsdataman.pas*») наименования полей записи «*TDataИдентификаторТаблицы*» («*TDataHotels*») будут иметь названия, совпадающие с введёнными идентификаторами полей.

Дескрипторы полей (*Display label*) – это заголовки столбцов на компоненте *WxGrid* модуля «*FrmИдентификаторТаблицы*» («*frmhotels.pas*» и «*frmhotels.lfm*»). Кроме того, они будут отображаться напротив компонентов ввода-вывода в диалоге, предназначенном для добавления и редактирования записей таблицы.

Дескриптор поля заносится в константу «*fd*ИдентификаторПоля» («*fdCityName*») (префикс *fd* является сокращением от *field descriptor*) модуля «ПРИЛОЖЕНИЕ*lng.pas*» («*travelslng.pas*»).

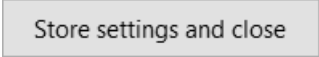
В дальнейшем, под именами таблиц и полей будут пониматься именно их идентификаторы.

После выбора необходимых настроек необходимо нажать кнопку



для их сохранения. Если требуется выйти из формы после

сохранения настроек, то необходимо использовать кнопку



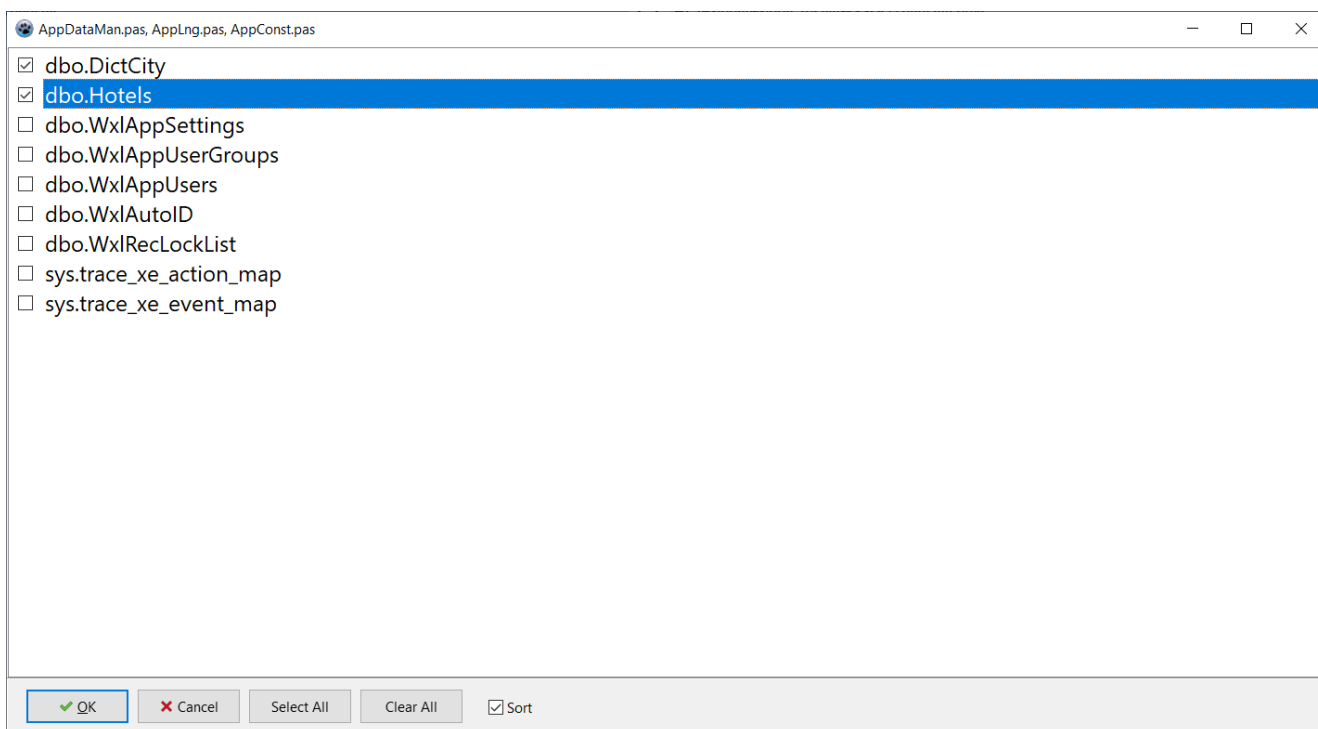
. Настройки сохраняются в следующей директории: «Директория *wxlgen.exe\wxlgen_result_zeos\СУБД\БД\settings*». Например, для предыдущего примера этот путь будет «*C:\wxlgen\wxlgen_result_zeos\MSSQL\Tourism\settings*». Имена файлов настроек: «*settings.wxt*», «*fields.wxt*», «*tables.wxt*».

4) После указания настроек можно приступить непосредственно к генерации исходных текстов и ресурсов WXL-приложения, а также приступить к генерации необходимых SQL-скриптов. Файлы складываются в директории назначения вида «Директория *wxlgen.exe\wxlgen_result_zeos\СУБД\БД*» («*C:\wxlgen\wxlgen_result_zeos\MSSQL\Tourism*»).

4.1) «SQL, *AppRecLock.inc*, *AppNextID.inc*» – в директории назначения появляется файл «*wxlgensql.sql*», также возможно появление файлов «ПРИЛОЖЕНИЕ*nextid.inc*» и «ПРИЛОЖЕНИЕ*relock.inc*». Для предыдущего примера это «*travelsnextid.inc*» и «*travelsrelock.inc*». На содержимое файла «*wxlgensql.sql*» влияют опции настроек «*Generate AppUserGroups, AppUsers support (PAS, SQL)*», «*Generate AppSettings support (PAS, SQL)*», «*Generate GetNextAutoID support (PAS, SQL)*» и «*Generate RecLock, RecUnlock support (PAS, SQL)*». При их отсутствии сгенерированный файл «*wxlgensql.sql*» будет пустым. При наличии опции «*Generate AppUserGroups, AppUsers support (PAS, SQL)*» в файле «*wxlgensql.sql*» дополнительно появляются SQL-скрипты для создания таблиц «*WxlAppUserGroups*» и «*WxlAppUsers*». В данных таблицах определены первичные ключи, а для второй из них определён также уникальный индекс «*IX_WxlAppUsers_RegName*». При наличии опции

«*Generate AppSettings support (PAS, SQL)*» в файле «*wxlgenssql.sql*» дополнительно появляются SQL-скрипты для создания таблицы пользовательских настроек «*WxlAppSettings*». При наличии опции «*Generate GetNextAutoID support (PAS, SQL)*» в файле «*wxlgenssql.sql*» дополнительно появляются SQL-скрипты для создания таблицы «*WxlAutoID*» и хранимой на сервере процедуры «*WxlGetNextID*». Таблица имеет два поля «*FieldName*» и «*ID*», поле «*FieldName*» – первичный ключ. При наличии опции «*Generate RecLock, RecUnlock support (PAS, SQL)*» в файле «*wxlgenssql.sql*» дополнительно появляются SQL-скрипты для создания таблицы «*WxlRecLockList*» и хранимых на сервере процедур «*WxlRecLock*» и «*WxlRecUnlock*». Созданные генератором SQL-скрипты необходимо выполнить при помощи любого удобного клиента, позволяющего выполнять запросы в базе данных выбранной СУБД.

4.2) «*AppDataMan.pas, AppLng.pas, AppConst.pas*» – в директории назначения появляются файлы «ПРИЛОЖЕНИЕ*dataman.PAS*», «ПРИЛОЖЕНИЕ*lng.pas*», «ПРИЛОЖЕНИЕ*const.pas*» и «ПРИЛОЖЕНИЕ*const.inc*». Для предыдущего примера это «*travelsconst.inc*», «*travelsconst.pas*», «*travelsdataman.pas*» и «*travelslng.pas*». В сгенерированном файле «*travelsdataman.pas*» содержатся определения записей, соответствующих полям выбранных и вспомогательных таблиц, и указателей на них. Также в данном файле содержатся определения наборов данных для работы с заданными таблицами. Файлы «*travelsconst.inc*» и «*travelsconst.pas*» служат для поддержки работы с идентификаторами и дескрипторами. Файл «*travelslng.pas*» содержит используемые приложением константы. Вид диалога для выбора таблиц для пунктов 4.2–4.5 показан на рисунке:



4.3) «*AppDataMod.pas*» – в директории назначения появляются файлы «ПРИЛОЖЕНИЕ*datamod.pas*» и «ПРИЛОЖЕНИЕ*datamod.lfm*». Для предыдущего примера это «*travelsdatamod.pas*» и «*travelsdatamod.lfm*». В сгенерированном *PAS*-файле содержится определение класса *TПРИЛОЖЕНИЕDataModule (TTravelsDataModule)* для централизованного управления невидимыми компонентами. Функциональность данной кнопки требует предварительного выполнения «*AppDataMan.pas, AppLng.pas, AppConst.pas*», так как используется файл «ПРИЛОЖЕНИЕ*dataman.pas*» («*travelsdataman.pas*»).

4.4) «*Forms.pas*» – в директории назначения появляются файлы «*frmИМЯТАБЛИЦЫ.pas*» и «*frmИМЯТАБЛИЦЫ.lfm*» (например, «*frmhotels.pas*» и «*frmhotels.lfm*»). Данный *PAS*-файл требует наличия файлов «ПРИЛОЖЕНИЕ*dataman.pas*» («*travelsdataman.pas*») и «*dlgИМЯТАБЛИЦЫ*» («*dlghotels.pas*» и «*dlghotels.lfm*»), поэтому требуется выполнить также «*AppDataMan.pas, AppLng.pas, AppConst.pas*» и «*Dialogs.pas*». Если используется несколько таблиц, то такие файлы появляются по одному на каждую таблицу. В сгенерированном *PAS*-файле содержится определение класса формы, выполняющей базовую функциональность *WXL*-приложения: отображение таблицы, обновление данных в таблице, вставка, редактирование

и удаление записи в таблице. Для запуска данной функциональности необходимо вызвать процедуру «*ProcessИМЯТАБЛИЦЫ*» (например, процедура «*ProcessHotels*», определённая в модуле «*FrmHotels*») из модуля «ПРИЛОЖЕНИЕ*main.pas*» («*travelsmain.pas*»).

4.5) «*Dialogs.pas*» – в директории назначения появляются файлы «*dlgИМЯТАБЛИЦЫ.pas*» и «*dlgИМЯТАБЛИЦЫ.lfm*» (например, «*dlghotels.pas*» и «*dlghotels.lfm*»). Данный *PAS*-файл требует наличия файлов «ПРИЛОЖЕНИЕ*dataman.pas*» («*travelsdataman.pas*») и «*frmИМЯТАБЛИЦЫ*» («*frmhotels.pas*» и «*frmhotels.lfm*»), поэтому требуется выполнить также «*AppDataMan.pas*, *AppLng.pas*, *AppConst.pas*» и «*Forms.pas*». Если используется несколько таблиц, то такие файлы появляются по одному на каждую таблицу. В сгенерированном *PAS*-файле содержится определение класса-диалога для введения и редактирования записей таблицы.

4.6) «*LOCAL TABLES.PAS*» – в директории назначения появляется файл «ПРИЛОЖЕНИЕ*etables.PAS*» («*travelstables.pas*»). Данный файл не требует наличия других сгенерированных файлов. Сгенерированный модуль содержит функциональность по работе с локальными таблицами.

4.7) «*App.lpr*, *AppMain.pas*, *AppCfg.pas*, *AppConnectData.pas*, *wxdefs.inc* ...» – в директории назначения появляются файлы:


- «ПРИЛОЖЕНИЕ*.lpi*» («*travels.lpi*»),
- «ПРИЛОЖЕНИЕ*.lpr*» («*travels.lpr*»),
- «ПРИЛОЖЕНИЕ*bounds.pas*» («*travelsbounds.pas*»),
- «ПРИЛОЖЕНИЕ*CFG.PAS*» («*travelscfg.pas*»),
- «ПРИЛОЖЕНИЕ*connectdata.pas*» («*travelsconnectdata.pas*»),
- «ПРИЛОЖЕНИЕ*localize.pas*» («*travelslocalize.pas*»),
- «ПРИЛОЖЕНИЕ*main.pas*» («*travelsmain.pas*»),
- «ПРИЛОЖЕНИЕ*main.lfm*» («*travelsmain.lfm*»),
- «*wxdefs.inc*».

Кроме того, при наличии соответствующих опций настроек появляются файлы «*dlgappsettings.pas*», «*dlgappsettings.lfm*», «*frmrelocklist.pas*» и «*frmrelocklist.lfm*».

Перед запуском этого пункта необходимо запустить «*AppDataMan.pas, AppLng.pas, AppConst.pas*» и «*AppDataMod.pas*», так как создаваемые данными кнопками файлы используется в главном модуле проекта. Основная задача данного пункта – создание проекта и главной формы приложения.

5) Пункт «*Display Labels*» представляет альтернативный метод доступа к функциональности по изменению идентификаторов и дескрипторов заданных таблиц.



6) Кнопка  служит для вызова анализатора запросов библиотеки *WXL* и для вызова вспомогательной информационной базы для выбранной базы данных на сервере.

7) Кнопка «*Additional*s» вызывает следующее всплывающее меню, содержащее дополнительную функциональность генератора:

7.1) «*Environment variables*». Отображение переменных среды.

7.2) «*AppCfg*». Отображение файла настроек генератора.

7.3) «*View TXT file*» позволяет просматривать текстовые файлы «*.txt*».

7.4) «*View DBASE table*» позволяет просматривать файлы «*.dbf*» посредством сетки.

7.5) «*View TBufDataSet file*» позволяет просматривать файлы *TBufDataSet* «*.bds*» посредством сетки.

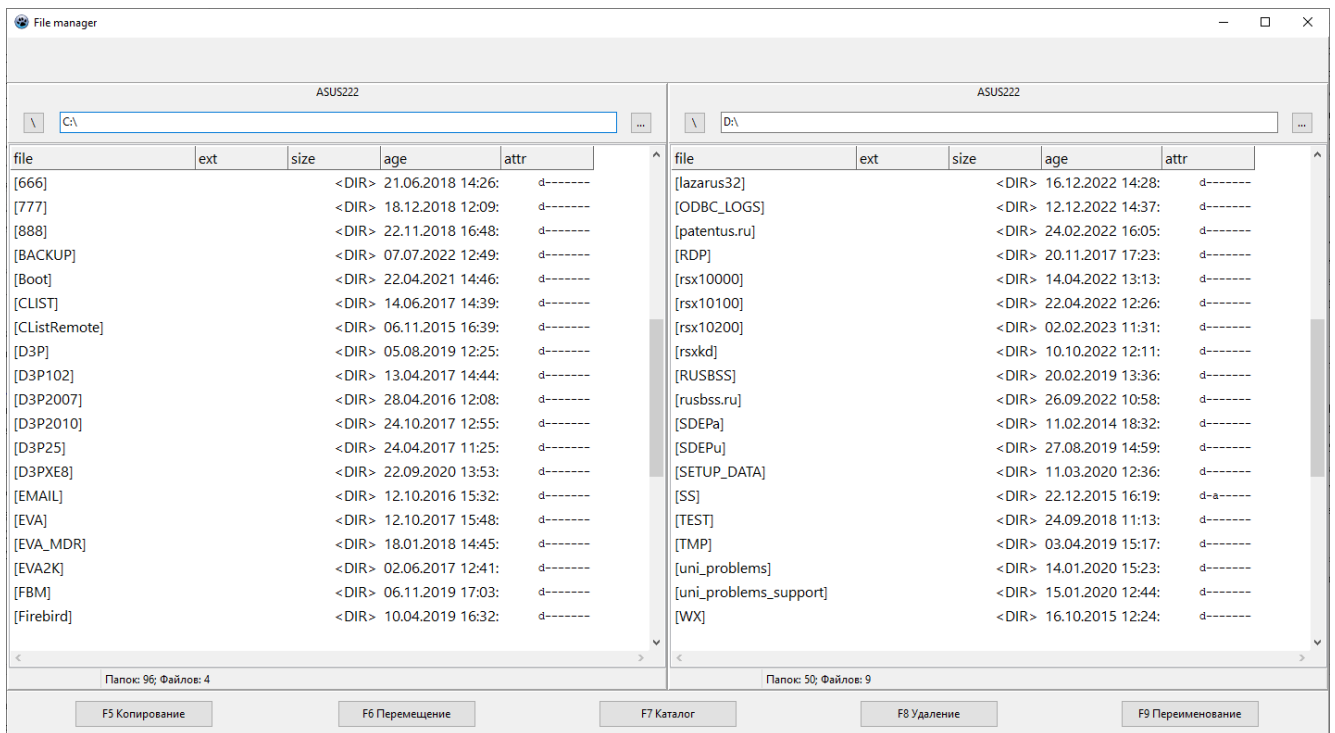
7.6) «*View TWxMemTable file*» позволяет просматривать файлы *TWxMemTable* «*.wxt*» посредством сетки.


7.7) «*Backup Database*». Сохранение базы данных для выбранной СУБД. Подробно данная функциональность описана в подразделе 10.4.3.

7.8) «*Backup Transaction Log*». Сохранение журнала транзакций для выбранной СУБД. Подробно данная функциональность описана в подразделе 10.4.3.

7.9) «*Load or Create file wxldbcon_zeos.lst*». Встроенная в генератор функциональность утилиты *wxldbconmgr* для работы с файлами соединения с выбранными СУБД.

7.10) «*File manager*». Встроенный двухпанельный файловый менеджер (аналогичный *Total Commander, Double Commander, ...*).



7.11) «*Debug Mode*». Включение или отключение режима отладки. Например, с её помощью можно посмотреть *SQL*-сценарии, обеспечивающие функциональность анализатора запросов, вызываемого по кнопке .

4.1.3 Генерация *SQL*-скриптов для объектов БД

Поставляемый вместе с библиотекой *WXL* генератор *WxlGen* позволяет создавать *SQL*-скрипты по созданию необходимых каждому *WXL*-приложению объектов баз данных:

- 1) *SQL*-скрипты для поддержания защищённого доступа к *WXL*-приложению. Создаются генератором *WxlGen* при наличии на форме настроек генератора крестика напротив «*Generate AppUserGroups, AppUsers support (PAS, SQL)*». Данная функциональность предназначена для создания специальных таблиц пользователей и групп пользователей для проверки прав доступа к *WXL*-приложению. Данная функциональность подробно рассмотрена в главе 11.
- 2) *SQL*-скрипты для поддержания таблицы пользовательских настроек. Создаются генератором *WxlGen* при наличии в форме настроек генератора крестика напротив «*Generate AppSettings support (PAS, SQL)*». Данная функциональность предназначена для создания специальной таблицы *WxlAppSettings*, содержащей информацию о пользовательских настройках.

- 3) *SQL*-скрипты для многопользовательского доступа к данным. Создаются генератором *WxlGen* при наличии в форме настроек генератора крестика напротив «*Generate RecLock, RecUnlock support (PAS, SQL)*». Данная функциональность предназначена для работы с так называемым «пессимистическим подходом для работы со многими пользователями» и подробно рассмотрена в разделе 10.6.
- 4) *SQL*-скрипты для генерации уникальных идентификаторов. Создаются генератором *WxlGen* при наличии в форме настроек генератора крестика напротив «*Generate GetNextAutoID support (PAS, SQL)*». Данная функциональность необходима, когда по каким-либо причинам недостаточно стандартных, предоставляемых многими СУБД типов *AUTOINCREMENT*, *TIMESTAMP* и *GUID* или их аналогов. Данная функциональность подробно рассмотрена в разделе 10.5.

4.1.4 Генерация исходных текстов и ресурсов *wxl*-приложений

Для генерации исходных текстов и ресурсов *WXL*-приложения используется генератор *WxlGen*. В первую очередь следует указать, что к работе непосредственно над приложением следует приступать только после проектирования самой базы данных и её объектов. Первые шаги по использованию генератора описаны при его описании. Укажем кнопки генератора, ориентированные на создание исходных текстов и ресурсов *WXL*-приложения:

- 1) «*AppDataMan.pas, AppLng.pas, AppConst.pas*» – в директории назначения появляются файлы «*ПРИЛОЖЕНИЕdataman.PAS*», «*ПРИЛОЖЕНИЕlng.pas*», «*ПРИЛОЖЕНИЕconst.pas*» и «*ПРИЛОЖЕНИЕconst.inc*». Для предыдущего примера это «*travelsconst.inc*», «*travelsconst.pas*», «*travelsdataman.pas*» и «*travelslng.pas*». В сгенерированном файле «*travelsdataman.pas*» содержатся определения записей, соответствующих полям выбранных и вспомогательных таблиц, и указателей на них. Также в данном файле содержатся определения наборов данных для работы с заданными таблицами. Файлы «*travelsconst.inc*» и «*travelsconst.pas*» служат для поддержки работы с идентификаторами и

дескрипторами. Файл «*travelslng.pas*» содержит используемые приложением константы.

- 2) «*AppDataMod.pas*» – в директории назначения появляются файлы «ПРИЛОЖЕНИЕ*datamod.pas*» и «ПРИЛОЖЕНИЕ*datamod.lfm*». Для предыдущего примера это «*travelsdatamod.pas*» и «*travelsdatamod.lfm*». В сгенерированном *PAS*-файле содержится определение класса *TПРИЛОЖЕНИЕDataModule* (*TTravelsDataModule*) для централизованного управления невизуальными компонентами. Функциональность данной кнопки требует предварительного выполнения «*AppDataMan.pas*, *AppLng.pas*, *AppConst.pas*», так как используется файл «ПРИЛОЖЕНИЕ*dataman.pas*» («*travelsdataman.pas*»).
- 3) «*Forms.pas*» – в директории назначения появляются файлы «*frmИМЯТАБЛИЦЫ.pas*» и «*frmИМЯТАБЛИЦЫ.lfm*» (например, «*frmhotels.pas*» и «*frmhotels.lfm*»). Данный *PAS*-файл требует наличия файлов «ПРИЛОЖЕНИЕ*dataman.pas*» («*travelsdataman.pas*») и «*dlgИМЯТАБЛИЦЫ*» («*dlghotels.pas*» и «*dlghotels.lfm*»), поэтому требуется выполнить также «*AppDataMan.pas*, *AppLng.pas*, *AppConst.pas*» и «*Dialogs.pas*». Если используется несколько таблиц, то такие файлы появляются по одному на каждую таблицу. В сгенерированном *PAS*-файле содержится определение класса формы, выполняющей базовую функциональность *WXL*-приложения: отображение таблицы, обновление данных в таблице, вставка, редактирование и удаление записи в таблице. Для запуска данной функциональности необходимо вызвать процедуру «*ProcessИМЯТАБЛИЦЫ*» (например, процедура «*ProcessHotels*», определённая в модуле «*FrmHotels*») из модуля «ПРИЛОЖЕНИЕ*main.pas*» («*travelsmain.pas*»).
- 4) «*Dialogs.pas*» – в директории назначения появляются файлы «*dlgИМЯТАБЛИЦЫ.pas*» и «*dlgИМЯТАБЛИЦЫ.lfm*» (например, «*dlghotels.pas*» и «*dlghotels.lfm*»). Данный *PAS*-файл требует наличия файлов «ПРИЛОЖЕНИЕ*dataman.pas*» («*travelsdataman.pas*») и «*frmИМЯТАБЛИЦЫ*» («*frmhotels.pas*» и «*frmhotels.lfm*»), поэтому требуется выполнить также «*AppDataMan.pas*, *AppLng.pas*, *AppConst.pas*» и «*Forms.pas*». Если

используется несколько таблиц, то такие файлы появляются по одному на каждую таблицу. В сгенерированном *PAS*-файле содержится определение класса-диалога для введения и редактирования записей таблицы.

- 5) «*LOCAL TABLES.PAS*» – в директории назначения появляется файл «ПРИЛОЖЕНИЕ*tables.PAS*» («*travelstables.pas*»). Данный файл не требует наличия других сгенерированных файлов. Сгенерированный модуль содержит функциональность по работе с локальными таблицами.
- 6) «*App.lpr, AppMain.pas, AppCfg.pas, AppConnectData.pas, wxdefs.inc ...*» – в директории назначения появляются файлы:
 - «ПРИЛОЖЕНИЕ*.lpi*» («*travels.lpi*»),
 - «ПРИЛОЖЕНИЕ*.lpr*» («*travels.lpr*»),
 - «ПРИЛОЖЕНИЕ*bounds.pas*» («*travelsbounds.pas*»),
 - «ПРИЛОЖЕНИЕ*CFG.PAS*» («*travelscfg.pas*»),
 - «ПРИЛОЖЕНИЕ*connectdata.pas*» («*travelsconnectdata.pas*»),
 - «ПРИЛОЖЕНИЕ*localize.pas*» («*travelslocalize.pas*»),
 - «ПРИЛОЖЕНИЕ*main.pas*» («*travelsmain.pas*»),
 - «ПРИЛОЖЕНИЕ*main.lfm*» («*travelsmain.lfm*»),
 - «*wxdefs.inc*».

Кроме того, при наличии соответствующих опций настроек появляются файлы «*dlgappsettings.pas*», «*dlgappsettings.lfm*», «*frmreclocklist.pas*» и «*frmreclocklist.lfm*». Перед запуском этого пункта необходимо запустить «*AppDataMan.pas, AppLng.pas, AppConst.pas*» и «*AppDataMod.pas*», так как создаваемые данными кнопками файлы используются в главном модуле проекта. Основная задача данного пункта – создание проекта и главной формы приложения.

В результате в директории с исполняемым файлом генератора в поддиректории с именем базы данных появятся следующие файлы:

- 1) «ПРИЛОЖЕНИЕ*.lpr*» («*travels.lpr*») – файл проекта сгенерированного типового *WXL*-приложения.
- 2) «ПРИЛОЖЕНИЕ*.lpi*» («*travels.lpi*») – файл описания проекта сгенерированного типового *WXL*-приложения.

- 3) «ПРИЛОЖЕНИЕ*bounds.pas*» («*travelsbounds.pas*») – вспомогательный файл для работы с ограничениями на размеры форм.
- 4) «ПРИЛОЖЕНИЕ*cfg.pas*» («*travelscfg.pas*») – вспомогательный файл по работе с расположением файлов, хранящих границы и масштабы форм.
- 5) «ПРИЛОЖЕНИЕ*connectdata.pas*» («*travelsconnectdata.pas*») – вспомогательный файл для поддержки работы по соединению с заданной СУБД.
- 6) «ПРИЛОЖЕНИЕ*main.pas*» («*travelsmain.pas*») – главный модуль сгенерированного типового *WXL*-приложения.
- 7) «*dlgИМЯТАБЛИЦЫ.pas*» («*dlghotels.pas*», ...) – количество файлов равно числу используемых таблиц. Реализация диалога добавления и исправления записи таблицы базы данных.
- 8) «*frmИМЯТАБЛИЦЫ.pas*» («*frmhotels.pas*», ...) – количество файлов равно числу используемых таблиц. Реализация базовой функциональности по работе с таблицей БД: вызов диалогов добавления и редактирования записей, удаление записей, отображение таблицы.
- 9) «ПРИЛОЖЕНИЕ*dataman.pas*» («*travelsdataman.pas*») – содержит определения записей, соответствующих строкам таблиц, и типизированных наборов данных. Это базовый модуль, в котором сосредоточены наиболее важные для приложения константы, переменные, типы, классы и функции.
- 10) «ПРИЛОЖЕНИЕ*const.pas*» («*travelsconst.pas*») – поддержка работы с идентификаторами и дескрипторами.
- 11) «ПРИЛОЖЕНИЕ*const.inc*» («*travelsconst.inc*») – поддержка работы с идентификаторами и дескрипторами.
- 12) «ПРИЛОЖЕНИЕ*lng.pas*» («*travelslng.pas*») – содержит используемые приложением константы и дескрипторы.
- 13) «ПРИЛОЖЕНИЕ*datamod.pas*» («*travelsdatamod.pas*») – для осуществления централизованного управления невидимыми компонентами.
- 14) «ПРИЛОЖЕНИЕ*tables.pas*» («*travelstables.pas*») – для работы с базами данных одноуровневой архитектуры.

- 15) «ПРИЛОЖЕНИЕ*localize.pas*» («*travelslocalize.pas*») – использование стандартного модуля *Lazarus* для локализации *Translations*.
- 16) «*dlgappsettings.pas*» – диалог для вставки записей в таблицу пользовательских настроек *WxlAppSettings*. Более подробную информацию об этой таблице можно получить из описания структуры типового *Wxl*-приложения в подразделе 4.1.5.
- 17) «ПРИЛОЖЕНИЕ*reclock.inc*» («*travelsreclock.inc*») – реализация функциональности по работе с таблицей блокировок «*WxlRecLockList*». Более подробно эта функциональность описана в разделе 10.6.
- 18) «*frmreclocklist.pas*» – реализация функциональности по работе с таблицей блокировок «*WxlRecLockList*». Более подробно эта функциональность описана в разделе 10.6.
- 19) «ПРИЛОЖЕНИЕ*nextid.inc*» («*travelsnextid.inc*») – реализация функциональности по генерации уникальных идентификаторов. Более подробно эта функциональность описана в разделе 10.5.
- 20) «*wxdefs.inc*» – стандартный файл с директивами условной компиляции *WXL*-приложения. Его подробное описание содержится в разделе 1.4.
- 21) «ПРИЛОЖЕНИЕ*main.lfm*» («*travelsmain.lfm*») – главная форма типового *WXL*-приложения.
- 22) «*dlgИМЯТАБЛИЦЫ.lfm*» («*dlghotels.lfm*», ...) – количество файлов равно числу используемых таблиц. Форма для диалога добавления и исправления записи таблицы базы данных.
- 23) «*frmИМЯТАБЛИЦЫ.lfm*» («*frmhotels.lfm*», ...) – количество файлов равно числу используемых таблиц. Форма для базовой функциональности по работе с таблицей БД: вызов диалогов добавления и редактирования записей, удаление записей, отображение таблицы.
- 24) «ПРИЛОЖЕНИЕ*datamod.lfm*» («*travelsdatamod.lfm*») – модуль данных (*TDataModule*).
- 25) «*dlgappsettings.lfm*» – форма для диалога вставки записей в таблицу пользовательских настроек *WxlAppSettings*. Более подробную информацию об

этой таблице можно получить из описания структуры типового *Wxl*-приложения в подразделе 4.1.5.

26) «*frmreclocklist.lfm*» – форма для работы с таблицей блокировок «*WxlRecLockList*».

При создании *WXL*-приложений рекомендуется в качестве файла проекта использовать «приложение.*lpr*» («*travels.lpr*»), изменяя его при необходимости. Для того чтобы вызвать формы, предназначенные для работы с конкретными таблицами базы данных, необходимо вызвать процедуру *ProcessИМЯТАБЛИЦЫ* модуля *FrmИМЯТАБЛИЦЫ* (*ProcessHotels* модуля *FrmHotels*). Например, на панель *PanelButtons* главной формы типового *WXL*-приложения можно поместить кнопку, а по её событию *OnClick* вызвать процедуру *FrmИМЯТАБЛИЦЫ.ProcessИМЯТАБЛИЦЫ* (*FRMHOTELS.ProcessHotels*).

4.1.5 Структура типового *wxl*-приложения

Типовое *WXL*-приложение, предназначенное для работы с *SQL*-серверами, имеет следующую структуру:

- 1) Главный модуль проекта «ПРИЛОЖЕНИЕ*main*» («*travelsmain*») с реализацией функциональности главной формы приложения по работе с меню и панелью кнопок. На панель кнопок сразу же рекомендуется добавить кнопки вызова форм для работы с таблицами. Описание базовых возможностей меню и панели кнопок будет приведено далее.
- 2) Модуль и форма с реализацией модуля данных. Используется следующее наименование модуля – «ПРИЛОЖЕНИЕ*datamod*» («*travelsdatamod*»). В нём содержится функциональность по соединению с базой данных.
- 3) Модуль для определения типизированных наборов данных и записей с данными для работы с таблицами. Используется следующее наименование модуля – «ПРИЛОЖЕНИЕ*dataman*» («*travelsdataman*»). Это базовый модуль, в котором сосредоточены наиболее важные для приложения константы, переменные, типы, классы и функции.
- 4) Модули и формы для работы с конкретными таблицами. Имеют названия «*frmИМЯТАБЛИЦЫ*» («*frmhotels*», ...). Содержат функциональность по

отображению данных при помощи *WxGrid*, кнопки для добавления, редактирования и удаления записей, кнопку для распечатки данных.

- 5) Модули и формы для диалогов по добавлению или редактированию записей. Имеют названия «*dlgИМЯТАБЛИЦЫ*» («*dlghotels*», ...).

Типовое *WXL*-приложение по работе с локальными базами данных включает в себя вместо модуля «ПРИЛОЖЕНИЕ*dataman*» («*travelsdataman*») модуль «ПРИЛОЖЕНИЕ*tables*» («*travelstables*») с такой же функциональностью.

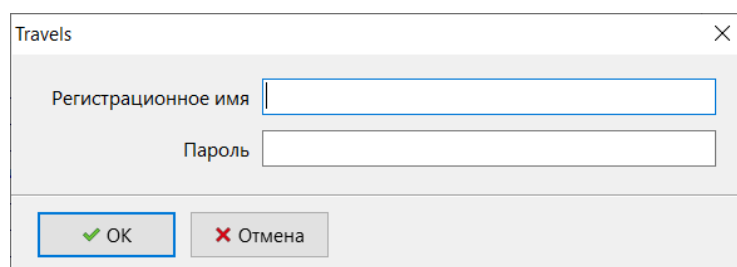
Написание типового *WXL*-приложения следует начинать с использования генератора *WxlGen*, который создаст весь каркас для него. Затем следует приступить к редактированию главного модуля и главной формы проекта. В частности, можно сразу же добавить кнопки для вызова функциональности по работе с конкретными таблицами. Когда все вызовы необходимых таблиц осуществлены, то можно приступить к редактированию остальных модулей и форм с целью обеспечения требуемой функциональности приложения баз данных.

Опишем каркас *WXL*-приложения, созданный генератором *WXLGEN*.

Главная форма только что созданного приложения имеет следующий вид:

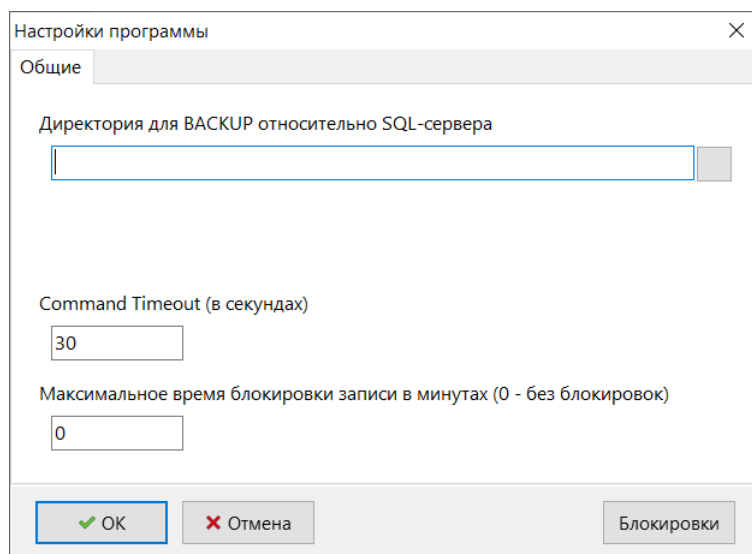


Чтобы получить данную форму в рабочем режиме необходимо пройти регистрацию для использования *WXL*-приложения. Форма регистрации имеет следующий вид:



Так как пользователи конкретного *WXL*-приложения неизвестны генератору *WXLGEN*, то существует регистрационное имя и соответствующий ему пароль,

которые можно использовать при первом использовании приложения. Это пользователь *ADMIN* с паролем *ADMIN*. Следует сменить пароль пользователя *ADMIN* при первой возможности. Если регистрация через *ADMIN* прошла успешно, то в первую очередь необходимо будет заполнить форму настроек программы:



Настройки программы

Общие

Директория для BACKUP относительно SQL-сервера


Command Timeout (в секундах)

30

Максимальное время блокировки записи в минутах (0 - без блокировок)

0

OK Отмена Блокировки

Требуется ввести директорию, в которой будут содержаться резервные копии базы данных или журнала транзакций, и нажать «ОК». Ввести директорию можно вручную, либо, нажав кнопку , расположенную справа от компонента ввода-вывода, вызвать соответствующий стандартный диалог выбора директории.

Следует ввести максимальное время блокировки записи в минутах, если необходимо использовать блокировки, и оставить ноль, если необходимо работать без использования блокировок.

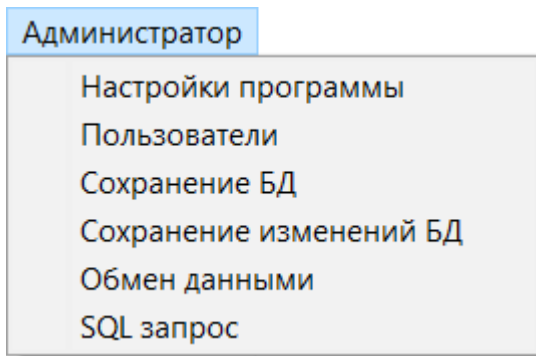
Command Timeout – это время, через которое, если команда всё ещё не выполнена, выполнение команды будет считаться неудачным. Это время должно быть указано в секундах.

К проекту присоединён модуль «*DlgAppSettings*», изменяя его и относящуюся к нему форму «*TDialogAppSettings*», можно добавлять дополнительные настройки, необходимые конкретному приложению. Рекомендуется, чтобы для этих целей создавались дополнительные вкладки, отличные от вкладки «Общие». Следует иметь в виду, что потребуется изменить также процедуры *SetData* и *GetData* класса *TDialogAppSettings* и изменить саму таблицу *WxlAppSettings*, предназначенную для сохранения выбранных настроек.

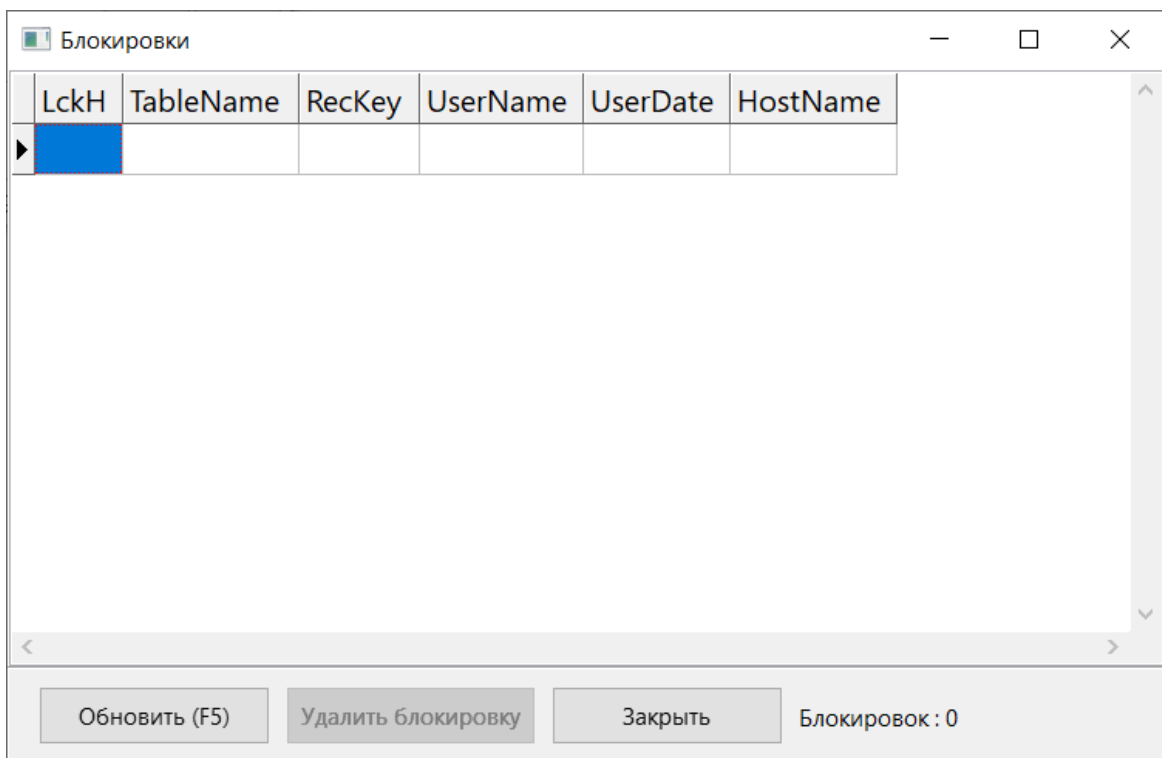
В дальнейшем данная форма не будет отображаться по умолчанию.

Рассмотрим возможности кнопок и пунктов меню в только что созданном WXL-приложении.

- 1) Левая кнопка на главной форме позволяет провести перерегистрацию для входа под именем другого пользователя.
- 2) Правая кнопка на главной форме позволяет выйти из приложения.
- 3) Вот как выглядит меню «Администратор»:



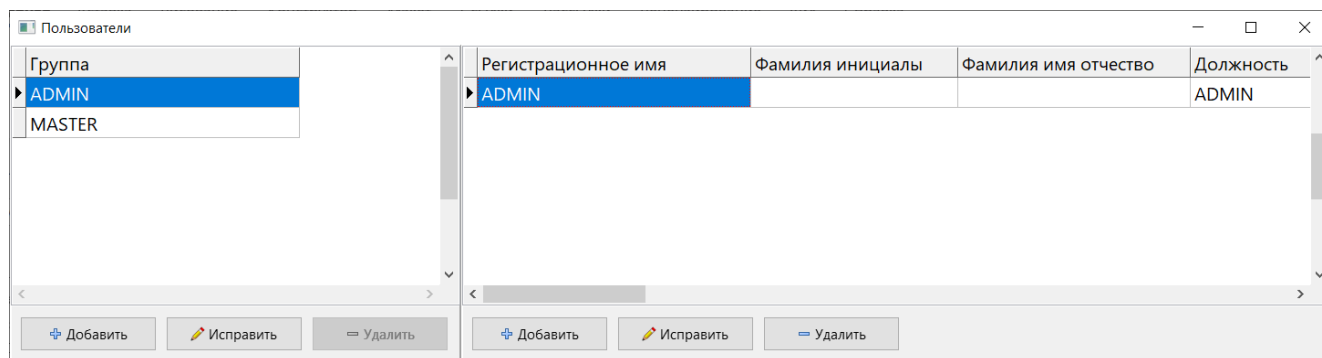
- 4) Первый пункт меню вызывает уже рассмотренную ранее форму настроек программы. Рассмотрим кнопку «Блокировки», расположенную на этой форме. При её нажатии появляется следующая форма с информацией о блокировках:



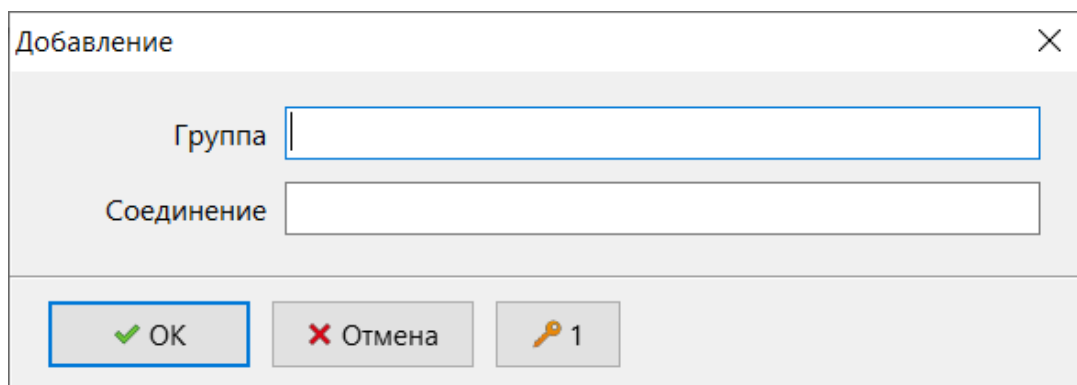
Путём нажатия кнопки «Обновить» или клавиши «F5» можно получить информацию о существующих на данный момент блокировках. При помощи кнопки «Удалить блокировку» при наличии прав администратора можно удалить выбранную блокировку. Данная функциональность предназначена для удаления

блокировок, которые уже не должны существовать, но по какой-то причине сохранились. Например, выключилось питание компьютера, и блокировка сохранилась несмотря на то, что должна была бы исчезнуть. При помощи кнопки «Заккрыть» можно закрыть форму с информацией о блокировках.


5) Пункт меню «Пользователи» позволяет ввести новые группы пользователей, а также добавить нового пользователя в заданную группу пользователей. Вот как выглядит форма для обеспечения заданной функциональности:

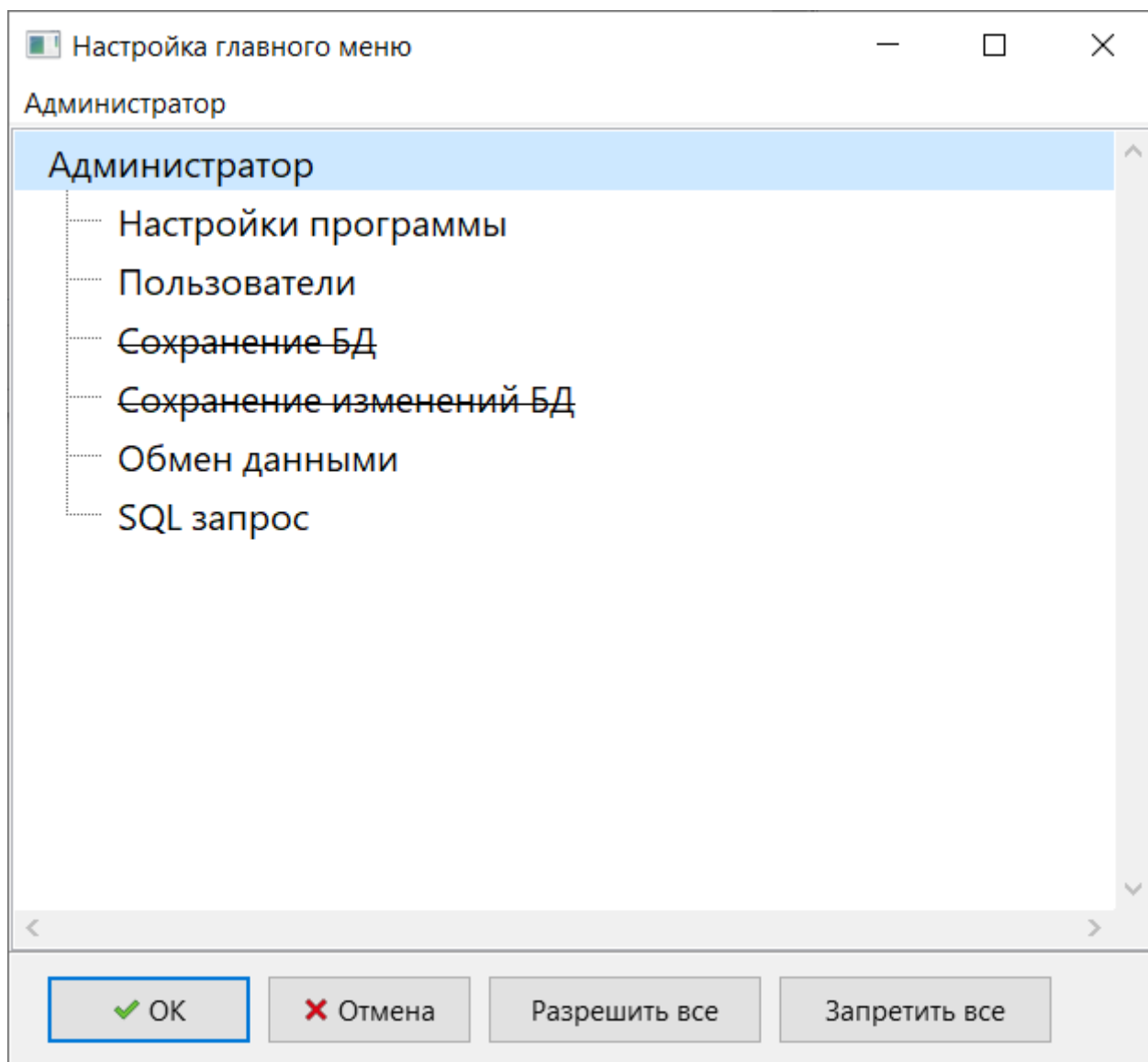


Соответственно, нажимая «Добавить» в левой части формы, можно добавить ещё одну группу пользователей *WXL*-приложения, а, нажав кнопку «Добавить» в правой части формы, можно добавить пользователя в выделенную группу пользователей слева. Соответственно, кнопки «Исправить» предназначены для редактирования существующих пользователей и групп пользователей *WXL*-приложения. Кнопки «Удалить» предназначены для удаления выбранной группы пользователей, либо пользователя внутри группы пользователей. Причём следует иметь в виду, что группу пользователей можно удалить в том, и только в том случае, если она не содержит пользователей. Вот как выглядит диалог добавления-редактирования новой группы пользователей (отличие в заголовке формы и в содержании компонента ввода-вывода, соответствующего метке «Группа»):





Нажатие на кнопку  позволяет выбрать доступные/недоступные пункты меню для заданной группы пользователей. Данная функциональность реализована в модуле *WxMenuTune*. Вот как выглядит в общем случае форма настройки меню модулем *WxMenuTune*:



Недоступные пункты меню зачёркнуты, доступные – не зачёркнуты. Кнопка «Разрешить все» делает доступными все пункты меню, кнопка «Запретить все» делает недоступными все пункты меню. Делать пункты меню доступными/недоступными можно при помощи клавиши «Insert».

Вот как выглядит диалог добавления-редактирования нового пользователя (отличие в заголовке формы и в содержании компонента ввода-вывода, соответствующего метке «Регистрационное имя»):

Добавление ×

Группа

Регистрационное имя


Пароль

Фамилия инициалы

Фамилия имя отчество

Должность

Телефон

Личная директория 


Стрелки как <TAB>

<ENTER> как <TAB>

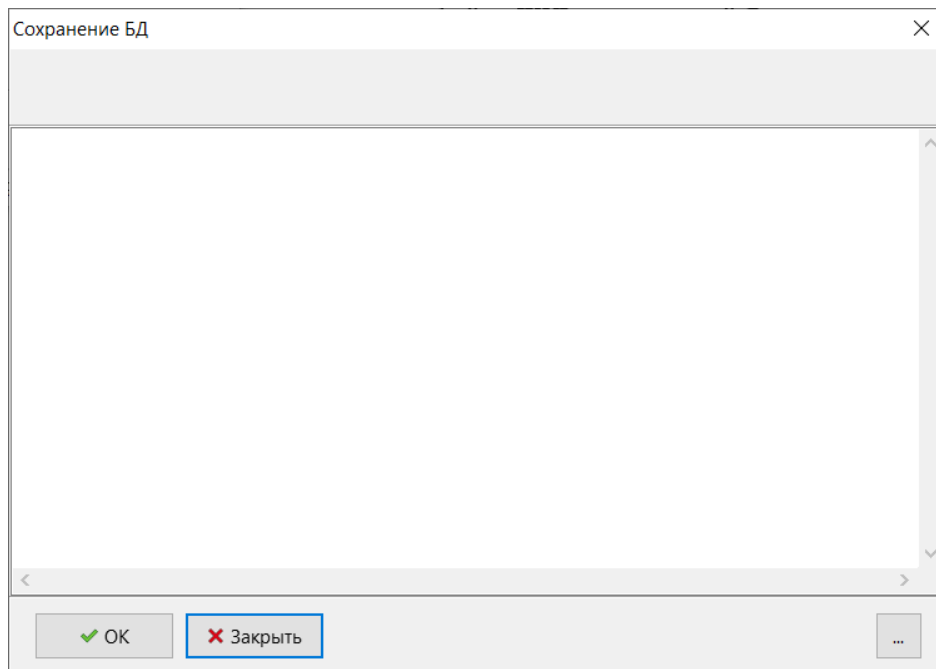
Поиск в полях со справочниками


Подсветка активного поля



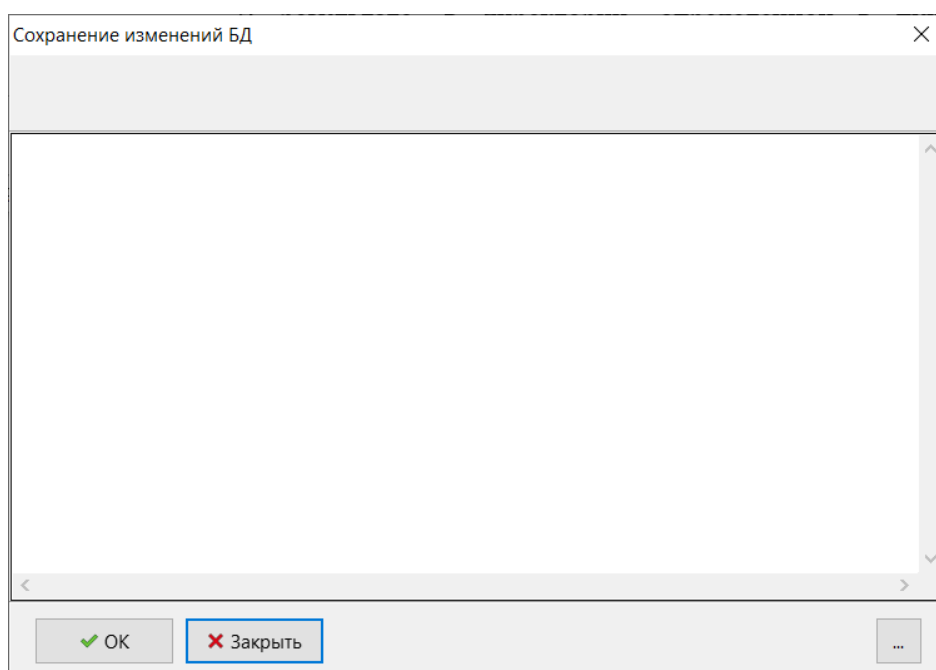
Нажатие на кнопку  позволяет выбрать доступные/недоступные пункты меню для заданного пользователя. Данная функциональность реализована в модуле *WxMenuTune*. Работа с этой функциональностью была описана ранее в этом же пункте. Описание опций «Стрелки как <TAB>», «<Enter> как <TAB>», «Поиск в полях со справочниками» и «Подсветка активного поля» содержится в описании интерфейса *WXL*-приложений в главе 5.


б) Пункт меню «Сохранение БД» вызывает форму, предназначенную для сохранения БД, реализованную в модуле *WxDbSQLBackUp*:



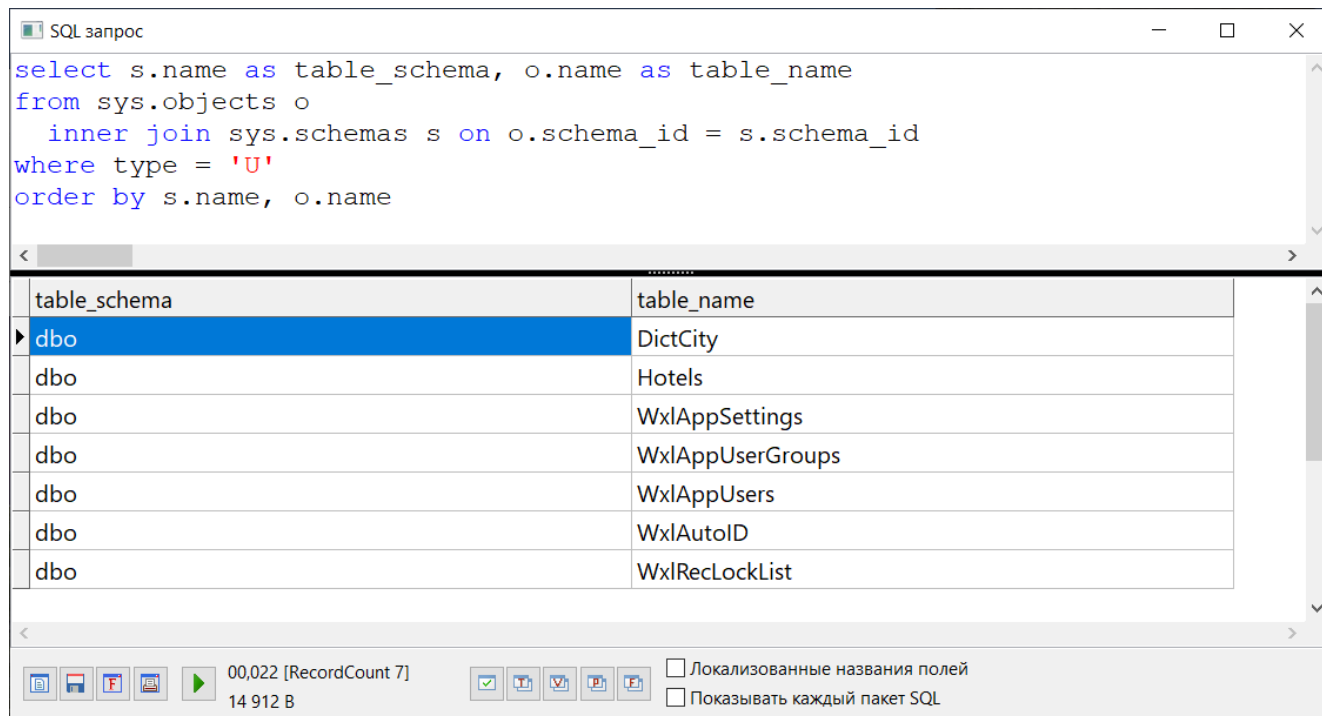
Кнопка «Закреть» отменяет выбор данного пункта меню, кнопка «ОК» проводит сохранение базы данных в выбранную в пункте меню «Настройки» директорию, кнопка  позволяет посмотреть историю резервных копирований. В результате в директории, определённой в пункте меню «Настройки», появляется резервная копия базы данных с именем «ИМЯБАЗЫДААННЫХ_DAT.BCK» («*TOURISM_DAT.BCK*»).

7) Пункт меню «Сохранение изменений БД» вызывает форму, предназначенную для сохранения журнала транзакций БД, реализованную в модуле *WxDbSQLBackUp*:






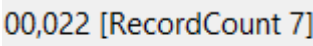
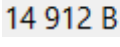







Кнопка «Закреть» отменяет выбор данного пункта меню, кнопка «ОК» проводит сохранение журнала транзакций базы данных в выбранную в пункте меню «Настройки» директорию, кнопка  позволяет посмотреть историю резервных копирований. В результате в директории, определённой в пункте меню «Настройки», появляется резервная копия журнала транзакций базы данных с именем «ИМЯБАЗЫДААННЫХ_LOG.BCK» («*TOURISM_LOG.BCK*»). В случае, если *RSC*-процедуры не установлены, то для резервного копирования журнала транзакций *Microsoft SQL Server* необходимо, чтобы *Recovery Model* не была *Simple*. Более подробно специфика осуществления резервного копирования для различных СУБД описана в подразделе 10.4.3.

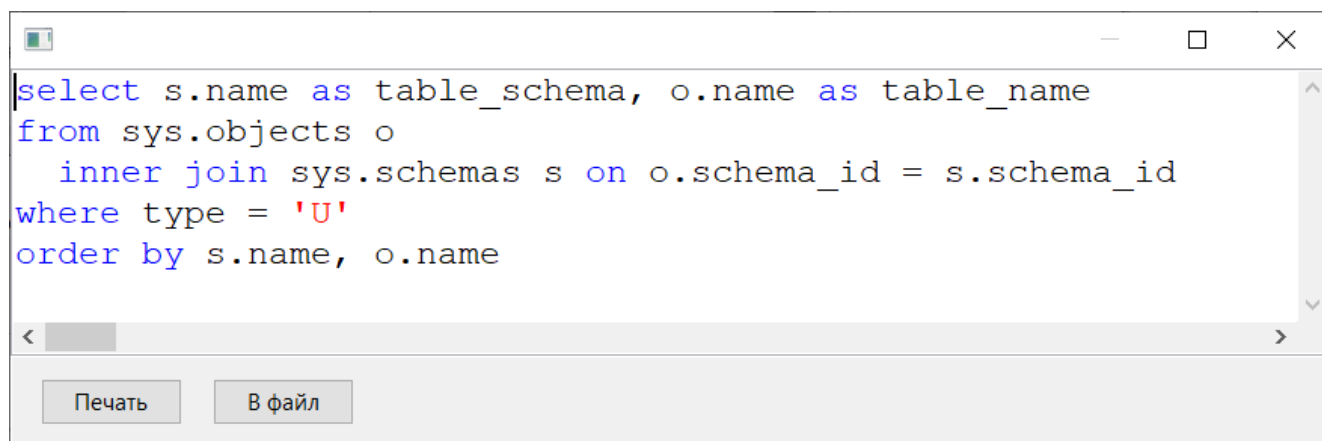
- 8) Пункт меню «Обмен данными» не содержит определённой функциональности. Он предназначен для обеспечения функциональности по репликации данных.
- 9) Пункт меню «*SQL* запрос» позволяет вызвать анализатор запросов библиотеки *WXL* с подключённой к нему вспомогательной информационной базой. Данная функциональность реализована в модуле *FrmSQLEx*. Вот как выглядит форма с анализатором запросов:



Как видно из рисунка, в верхней части формы пишется запрос, а в нижней части формы отображается результирующий набор данных. Для выполнения

запроса необходимо нажать клавишу «F5», либо кнопку . Кнопка  предназначена для загрузки запроса из файла с расширением «.sql». Кнопка  предназначена для сохранения запроса в файл с расширением «.sql». По соглашению, рекомендуется хранить такие файлы в директории *USERS* созданного типового *WXL*-приложения. Кнопка  позволяет менять шрифт, которым пишется запрос, путём вызова стандартного диалога пользователя по изменению шрифта. Кнопка  предназначена для распечатки текста. После кнопок анализатора содержится информация о времени выполнения запроса и числе затронутых записей:  . Правая часть панели содержит кнопки для использования информационной базы для выбранной базы данных. Кнопка  предназначена для просмотра опций базы данных. Кнопка  вызывает форму, предназначенную для просмотра списков имён таблиц. Путём нажатия правой кнопки мыши можно вызвать дополнительную функциональность по работе с выбранной таблицей. Кнопка  вызывает форму, предназначенную для просмотра списков имён представлений. Путём нажатия правой кнопки мыши можно вызвать дополнительную функциональность по работе с выбранным представлением. Кнопка  вызывает форму, предназначенную для просмотра списков хранимых на сервере процедур. Путём нажатия правой кнопки мыши можно вызвать дополнительную функциональность по работе с хранимыми на сервере процедурами. Кнопка  вызывает форму, предназначенную для просмотра списков хранимых на сервере функций. Путём нажатия правой кнопки мыши можно вызвать дополнительную функциональность по работе с хранимыми на сервере функциями. Галочка «Локализованные названия полей» используется для учёта используемых в приложении дескрипторов полей. В частности, для ранее рассмотренного примера запрос «select * from DictCity» вернёт набор данных с заголовками полей «ID», «Населённый пункт», «Актуально (0/1)».

Анализатор запросов позволяет использовать ключевое слово «*GO*» для разделения отдельных пакетов. Если дополнительно поставить галочку «Показывать каждый пакет *SQL*», то каждый пакет будет отображён в отдельном окне.



```
select s.name as table_schema, o.name as table_name
from sys.objects o
     inner join sys.schemas s on o.schema_id = s.schema_id
where type = 'U'
order by s.name, o.name
```

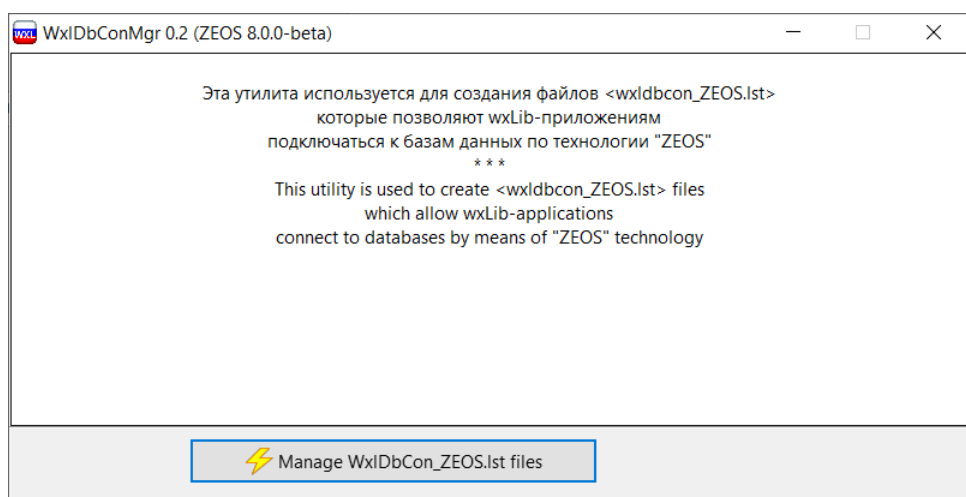
Печать В файл

Кнопка «Печать» предназначена для распечатки запроса, кнопка «В файл» – для его сохранения в файл.

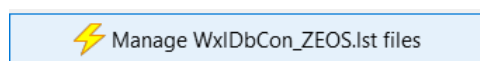
4.2 Утилита *wxldbconmgr*

4.2.1 Описание утилиты

Утилита *wxldbconmgr* предназначена для создания файлов, предназначенных для зашифрованного хранения информации о параметрах подключения приложений к базам данных. Существуют сборки утилиты для технологий доступа к данным *SqlDB* и *ZeosDBO*, для 32 и 64 бит. Внешний вид утилиты после запуска:

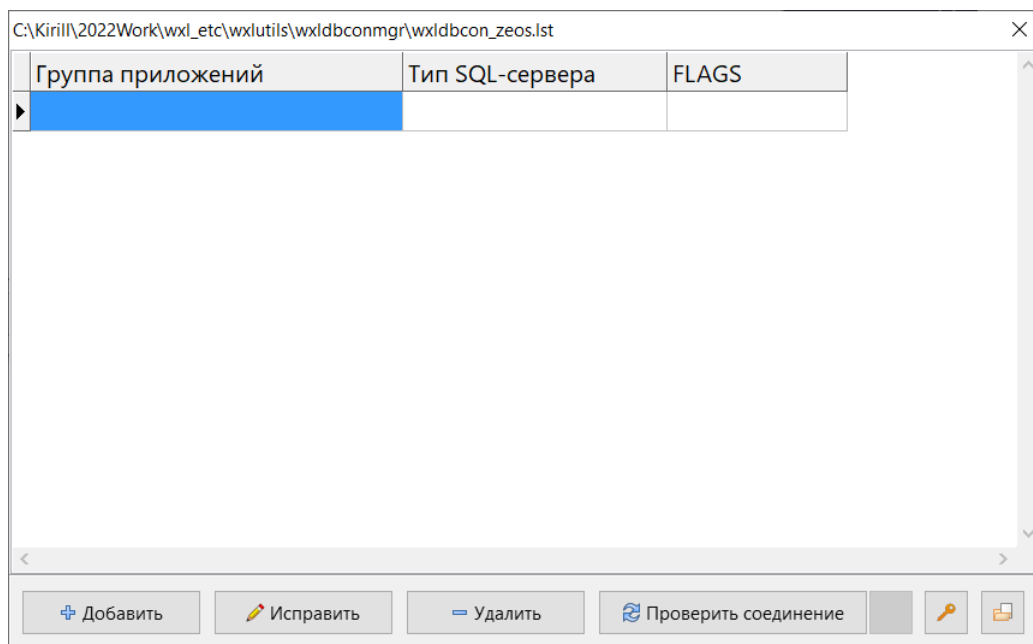


Для начала работы следует нажать единственную доступную кнопку

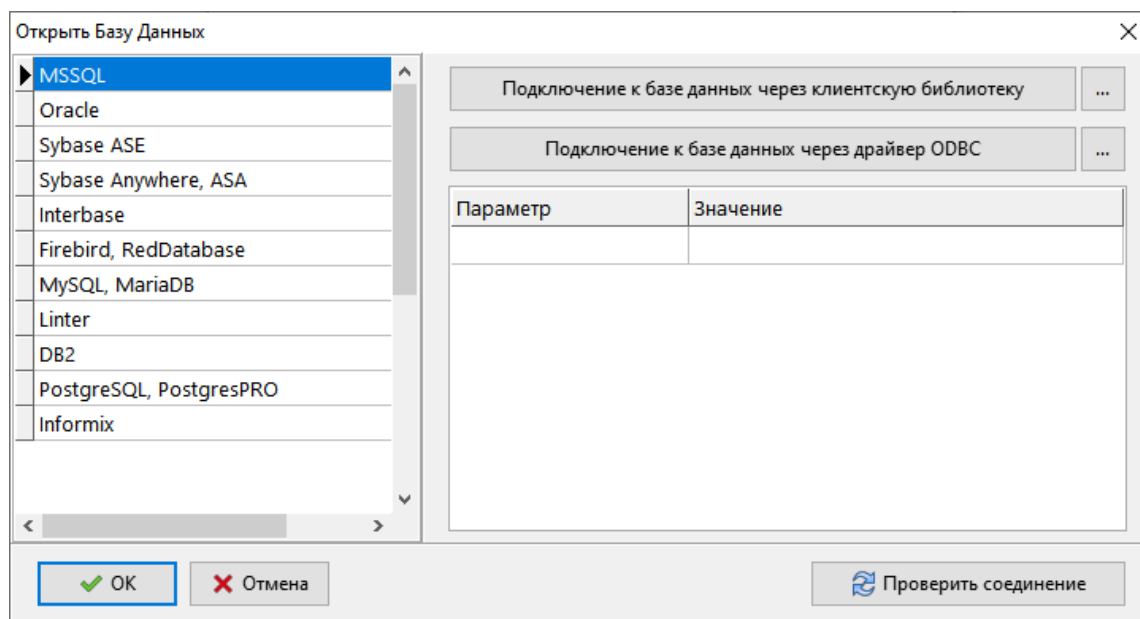


. После этого необходимо выбрать папку, в которой хранится *LST*-файл. Если *LST*-файл отсутствует в папке, то он будет создан, если

присутствует – загружен. Внешний вид формы для первого случая показан на рисунке:



Кнопка служит для добавления новой строки соединения. Форма для добавления выглядит следующим образом:



При помощи сетки в левой части формы можно выбрать тип *SQL*-сервера: «*MSSQL*», «*Interbase*», «*Firebird*» или «*PostgreSQL*». Доступ через *ODBC* при помощи кнопки «Подключение к базе данных через драйвер *ODBC*» доступен для любой СУБД, доступ через клиентскую библиотеку при помощи кнопки «Подключение к базе данных через клиентскую библиотеку» также возможен для всех четырёх СУБД. Далее необходимо выбрать способ соединения с базой данных

на сервере, нажав соответствующую кнопку. Внешний вид диалога соединения при выборе «Подключение к базе данных через клиентскую библиотеку» на примере *Microsoft SQL Server*:

MSSQL

Provider mssql

Server

Database

User

Password


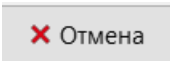
Port

Client CodePage

Library Location ...

Use Metadata

OK Отмена

Кнопкой  можно подтвердить введённые параметры соединения, кнопкой  можно отказаться от них. Внешний вид формы после подтверждения введённых параметров:

Добавить

МSSQL

Interbase

Firebird

PostgreSQL

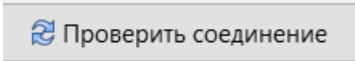
Подключение к базе данных через клиентскую библиотеку


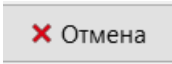
Подключение к базе данных через драйвер ODBC

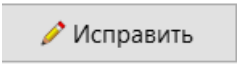
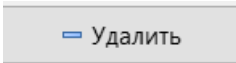


Параметр	Значение
Provider	mssql
Server	MSSQL2016
Database	OKO777
User	sa
Password	#####
codepage	UTF-8
LibLocation	libsybdb-5.dll

Группа приложений OKO777

OK Отмена Проверить соединение

Необходимо указать группу приложений, для которой будет использоваться строка соединения. Кнопка  служит для проверки

соединения. Кнопкой  можно подтвердить добавление новой строки соединения, кнопкой  можно отказаться от него.

Кнопка  предназначена для исправления ранее введённой строки соединения. Кнопка  предназначена для удаления ранее введённой строки соединения. Кнопка  служит для проверки соединения. Кнопка  предназначена для просмотра параметров соединения в текстовом виде.

4.2.2 Использование файлов *wxldbcon_*.lst* в *wxl*-приложениях

В комплект поставки инструментальной библиотеки *wxLib* для *FPC+Lazarus* входят утилиты *wxlGen*, позволяющие генерировать исходный код приложений баз данных:

- *wxlgen_sqldb* (используется технология *SqlDB*),
- *wxlgen_zeos* (используется технология *ZeosDBO*).

Эти утилиты, а также утилиты

- *wxldbconmgr_sqldb* (используется технология *SqlDB*),
- *wxldbconmgr_zeos* (используется технология *ZeosDBO*),

позволяют создавать файлы *wxldbcon_sqldb.lst* / *wxldbcon_zeos.lst*, в которых в криптографированном виде хранится информация о параметрах подключения приложений к базам данных.

Для ведения файлов "*wxldbcon_*.lst*" используйте:

- в *wxlGen* кнопка "Additional" / позиция меню «Load or Create file *wxldbcon_zeos.lst*»,
- в *wxldbconmgr* кнопка «Manage *wxldbcon_*.lst* files».

Если Вы сгенерировали исходный код приложения утилитой *wxlGen*, то Вам доступен простой способ настройки параметров подключения приложений к базе данных, не требующий внесения изменений в исходный код приложений.

Если рядом с wxl-приложением, код которого сгенерирован утилитой *wxlGen*, положить файл *wxldbcon_sqldb.lst* или *wxldbcon_zeos.lst*, приложение будет читать из него параметры подключения к БД.

Ниже приведён фрагмент кода главной формы приложения, сгенерированного утилитой *wxlGen*, в котором показан механизм использования файлов *wxldbcon_*.lst*

```
function T{MyWxlApp}MainForm.BeginWork: Boolean;
var
  ConnectData: WXTYPES.TWxSQLConnectData;
begin
  Result := False;
  .....
  и т.д.
  .....
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// Здесь выполняется попытка загрузки из файла wxldbcon_*.lst информации
// о параметрах подключения приложения к БД и попытка подключения к БД:
  {MyWxlApp}CONNECTDATA.LoadSQLConnectData;
  {MyWxlApp}CONNECTDATA.GetSQLConnectData(ConnectData);
{MyWxlApp}DATAMOD.{MyWxlApp}DataModule.DataBaseConnect(ConnectData);
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

  .....
  и т.д.
  .....
end;
```

В модуле *{MyWxlApp}ConnectData.pas*, сгенерированном утилитой *wxlGen*, объявлена константа:

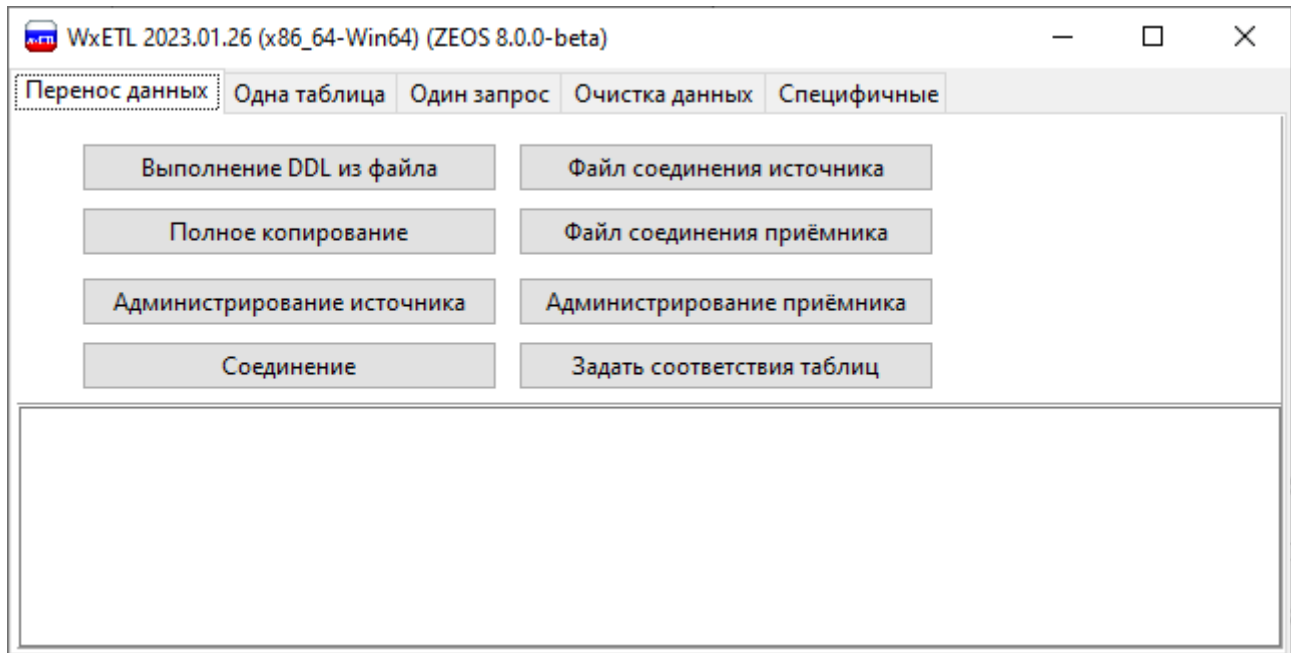
```
const
  GAppGroup = '<имя_группы_в_файле_wxldbcon_*.lst>';
```

Именно с этим значением *GAppGroup* Ваше приложение будет искать запись в *wxldbcon_*.lst*.

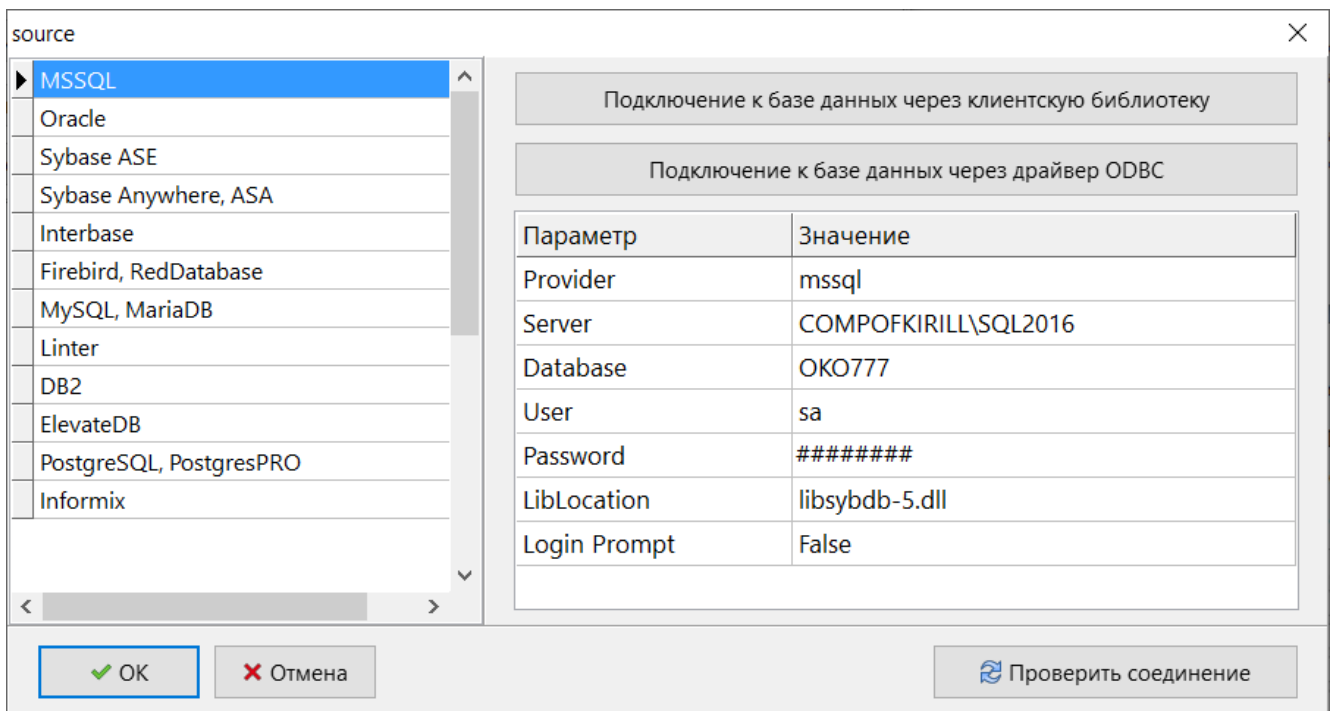
4.3 Утилита *wxetl*

Утилита *wxetl* представляет собой средство для переноса данных между таблицами баз данных, которые могут находиться под управлением различных СУБД. Соответствия между таблицами источника и приёмника задаются при помощи файла «*settings.ini*». В программе используется пять вкладок:

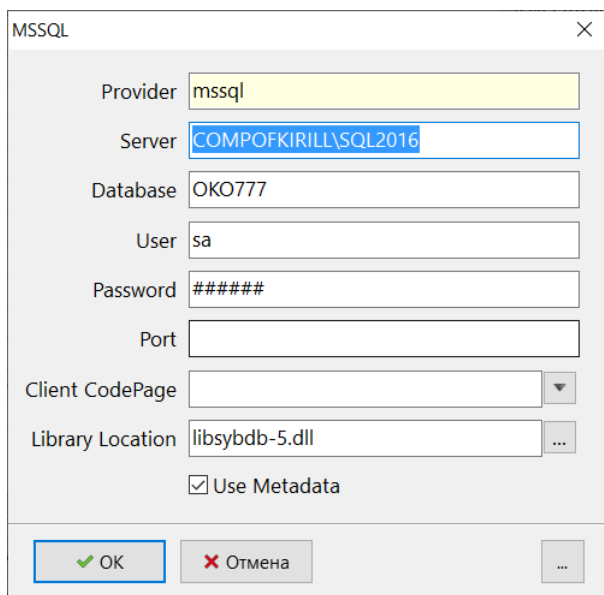
1) Перенос данных.



1.1) При первом запуске необходимо настроить соединение с источником (например, *Microsoft SQL Server*) и приёмником (например, Ред База Данных). Это можно сделать либо при помощи кнопок на вкладке, либо при помощи редактирования файлов выбранной технологии доступа к данным *wxldbcon_sqldb.lst*, *wxldbcon_zeos.lst* (например, с помощью утилит *wxlgen* или *wxldbconmgr*). Пример диалога при использовании кнопок «Файл соединения источника» и «Файл соединения приёмника».



Вид внутреннего диалога, вызываемого при нажатии кнопки «Подключение к базе данных через клиентскую библиотеку»:



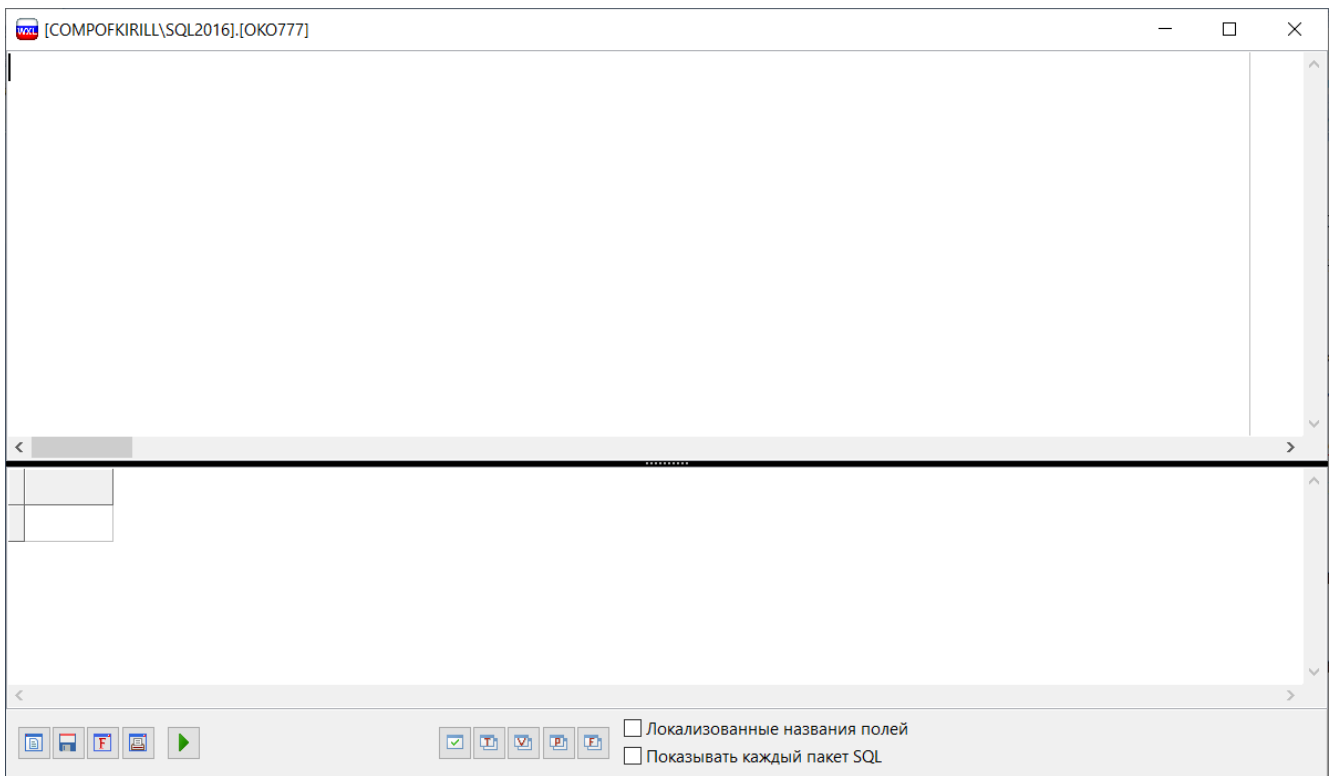
Само соединение с базами данных источника и приёмника выполняется по нажатию клавиши «Соединение». Если в файле «*wxetl.ini*» указана опция *AutoConnect=true*, то соединение будет установлено автоматически при старте программы.

1.2) Для переноса данных используется кнопка «Полное копирование». В текущей реализации данные копируются для всех соответствий запросов к таблицам источника и таблиц приёмника, заданных в файле «*correspondence.dat*». Пример лога содержится в текстовом файле «Пример лога.txt». Таблицы приёмника должны быть предварительно очищены. Для хранения соответствий используется набор данных в виде таблицы в памяти *TWxMemDataSet*.

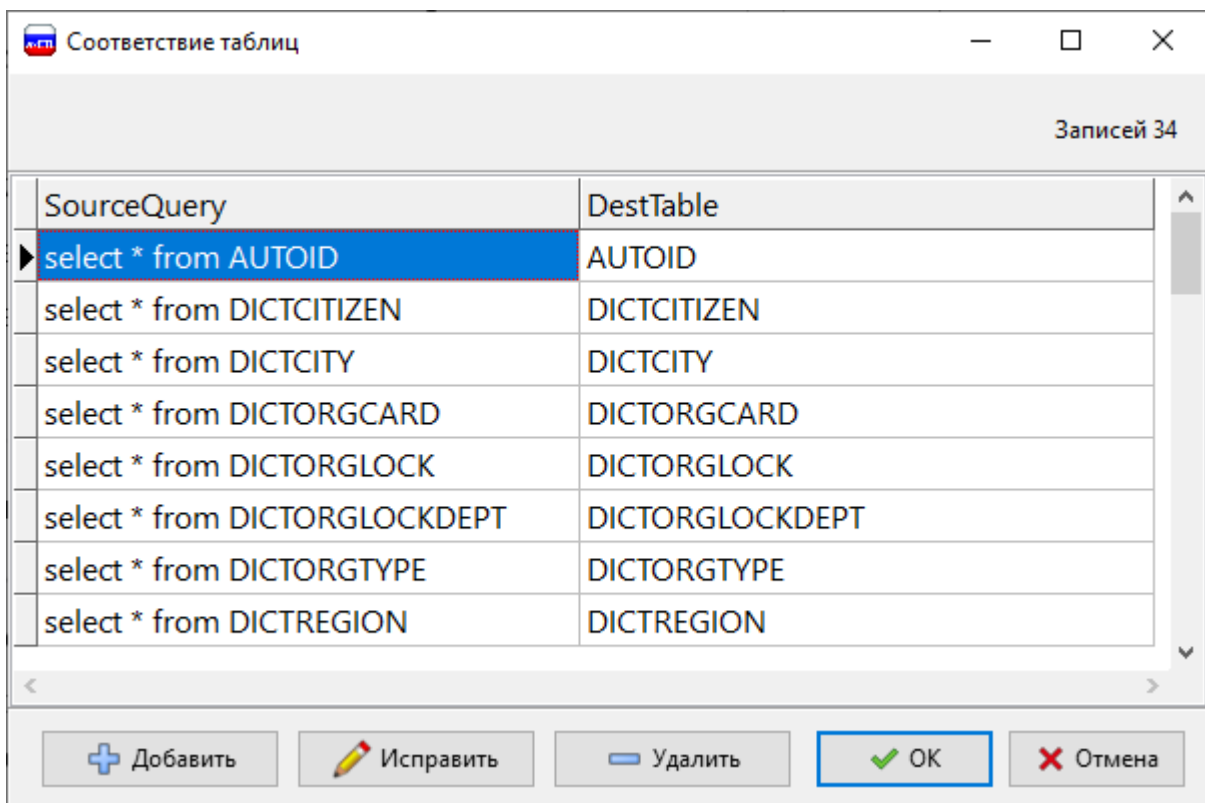
1.3) Кнопка «Выполнение *DDL* из файла» служит для выполнения произвольных сценариев, которые не возвращают данные. Скрипты необходимо разместить в файле «*createobjects.sql*». В качестве разделителя служит ключевое слово *go*, написанное произвольным регистром на отдельной строке.

1.4) Кнопка «Администрирование источника» вызывает стандартную *WXL*-форму выполнения сценариев для источника.

1.5) Кнопка «Администрирование приёмника» вызывает стандартную *WXL*-форму выполнения сценариев для приёмника.



1.6) Кнопка «Задать соответствия таблиц» вызывает форму для отображения и редактирования списка соответствий запросов к таблицам источника и таблиц приёмника.



Кнопки «Добавить» и «Исправить» позволяют вызвать диалог для добавления или исправления выбранного соответствия.

Исправление

Источник

Назначение

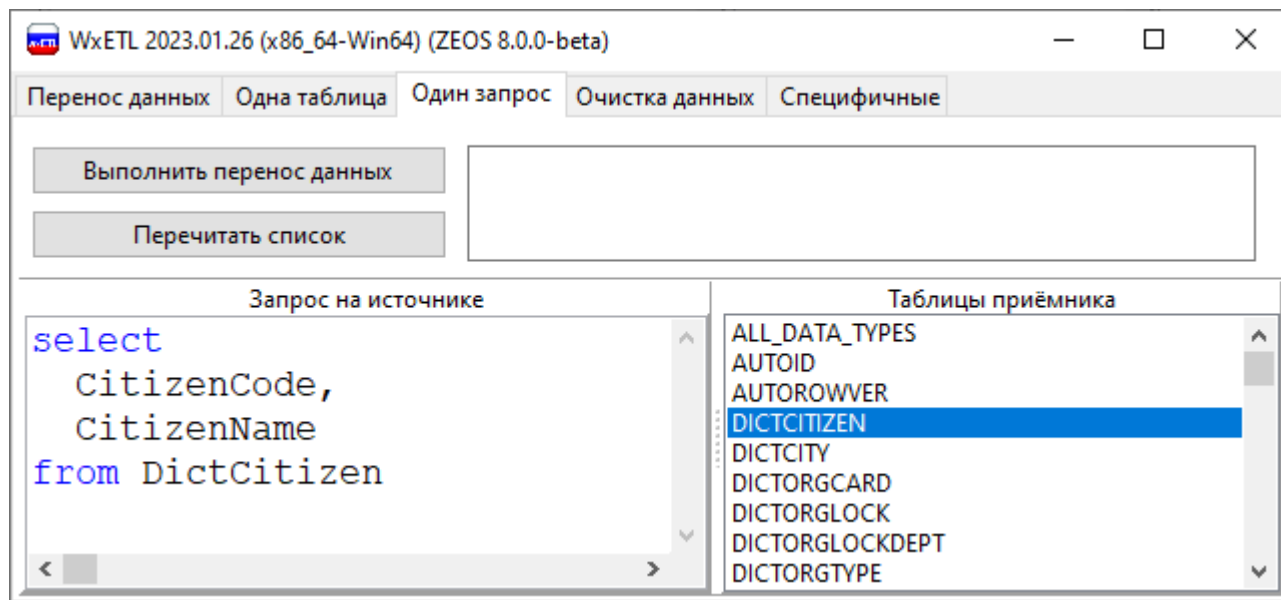
Кнопка «Удалить» позволяет удалить текущее соответствие. Кнопка «OK» закрывает форму, сохраняя изменения в файл «correspondence.dat». Кнопка «Отмена» позволяет выйти, не сохраняя внесённые изменения.

2) Одна таблица.

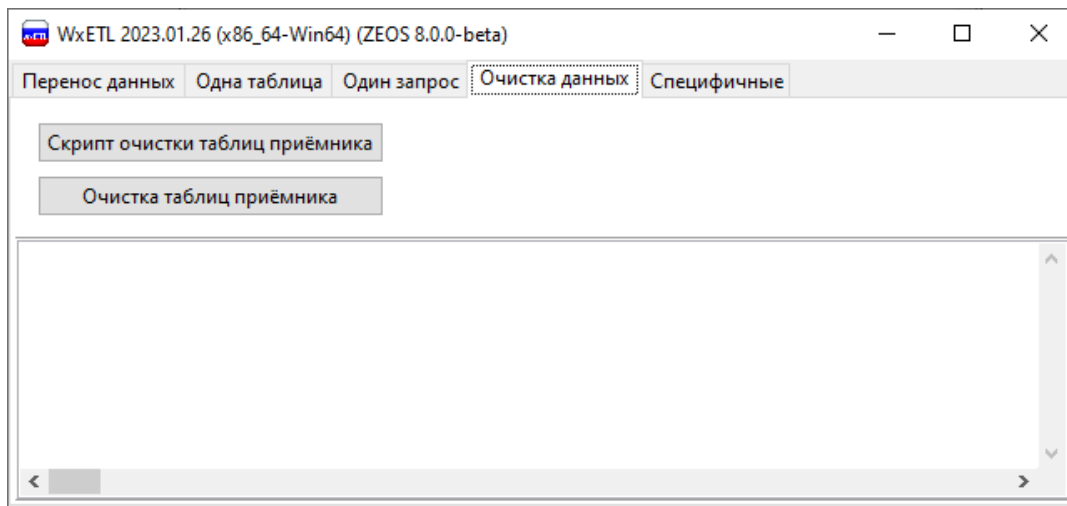
WxETL 2023.01.26 (x86_64-Win64) (ZEOS 8.0.0-beta)

Таблицы источника	Таблицы приёмника
dbo.AutoID	AUTOID
dbo.DictCitizen	AUTOROWVER
dbo.DictCitizen_SSHISTORY	DICTCITIZEN
dbo.DictCity	DICTCITY
dbo.DictOrgCard	DICTORGCARD
dbo.DictOrgCard_SSHISTORY	DICTORGLOCK
dbo.DictOrgLock	DICTORGLOCKDEPT
dbo.DictOrgLockDept	DICTORGTYP
dbo.DictOrgLockDept_SSHISTORY	DICTREGION

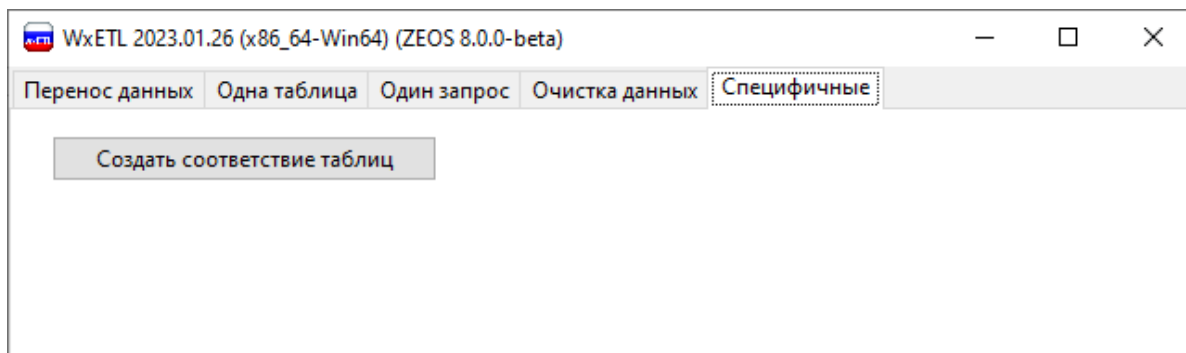
- 2.1) Кнопка «Копирование 1 таблицы». Таблицы выбираются из двух списков (слева - источник, справа - приёмник). Списки заполняются при запуске приложения или по нажатию кнопки «Перечитать списки».
- 2.2) Кнопка «Перечитать списки» служит для обновления информации в двух списках (изменение состава таблиц в БД, изменение строк соединения).
- 3) Один запрос



- 3.1) Кнопка «Выполнить перенос данных». Набор данных заполняется на основе запроса, который указан в левой части вкладки. Эти данные вставляются в таблицу, выбранную при помощи списка в правой части вкладки. Список заполняется при запуске приложения или по нажатию кнопки «Перечитать список». Столбцы набора данных и таблицы считаются соответствующими друг другу, если имеют одинаковые имена. Поэтому при использовании в запросе вычисляемых столбцов (например, при использовании функций преобразования типов данных) необходимо давать им псевдонимы.
- 3.2) Кнопка «Перечитать список» служит для обновления информации в списке в правой части вкладки (изменение состава таблиц в БД, изменение строк соединения).
- 4) Очистка данных.



- 4.1) Кнопка «Скрипт очистки таблиц приёмника» служит для получения скрипта очистки ВСЕХ таблиц приёмника для последующего запуска в любой среде разработки на языке *SQL* (например, с помощью *rsxadm* или *rssadm*).
- 4.2) Кнопка «Очистка таблиц приёмника» служит для очистки ВСЕХ таблиц приёмника непосредственно из программы (в том числе тех, которые не будут копироваться).
- 5) Специфичные.



- 5.1) Кнопка «Создать соответствия таблиц» позволяет заполнить соответствия таблиц в файле «correspondence.dat» автоматически, считая таблицы с одинаковыми именами соответствующими друг другу.

Утилита *wxetl* использует набор библиотек *FreeTDS* для доступа к *Microsoft SQL Server*. Их поведение зависит от файла настроек "*freetds.conf*". При отсутствии этого файла используются значения по умолчанию, которые могут не подходить для используемой версии СУБД. В операционной системе *Windows* местонахождение файла "*freetds.conf*" определяется значением переменной среды (*environment variable*) *FREETDSCONF*. Пример добавления переменной среды через *GUI Windows 10*: найти значок "Мой компьютер", правая кнопка мыши,

Свойства, Дополнительные параметры системы, Свойства системы, Дополнительно, Переменные среды, "выбрать либо переменные среды пользователя, либо системные переменные", установка нужной переменной. Пример добавления переменной среды через командную строку: `setx FREETDSCONF ПУТЬ\freetds.conf`. Пример файла "`freetds.conf`":

```
[global]
tds version = 8.0
client charset = UTF8
#dump file = freetds.log
```

Символ решётка (#) в файле настроек обозначает комментарий. Указание версии *TDS* обязательно, иначе возникнут проблемы при взаимодействии с типами данных, которые появились, начиная с *Microsoft SQL Server 2005* и выше (например, *varchar(max)* и *nvarchar(max)*). Параметр "*dump file*" может быть полезно установить при возникновении проблем взаимодействия с *SQL*-сервером. Не стоит устанавливать его в режиме штатной работы, так как его использование приводит к значительному замедлению работы клиентских приложений. Файл дампа и версия *FreeTDS* могут быть также заданы через переменные среды *TDSDUMP* и *TDSVER*. Значения переменных среды имеют приоритет перед параметрами, заданными внутри файла "`freetds.conf`".





5. Основы графического интерфейса wxl-приложений

Графический интерфейс WXL-приложений составлен в соответствии с требованиями спецификации «*CUA*» (*Common User Access*), предложенной фирмой *IBM* в 1984 году и ставшей стандартом в области интерфейса компьютерных приложений. «*CUA*» представляет собой часть рекомендуемых стандартов фирмы *IBM* для обработки информации «*SAA*» (*System Application Architecture*).

Любые пользователи WXL-приложений могут быть отнесены к одной из двух групп пользователей: к администраторам или операторам. Администраторы – это пользователи, которые имеют права на выполнение всех операций в программе. В частности, они могут выполнять задачу администрирования прочих пользователей (администраторов и операторов). Операторы – это пользователи, которые не относятся к категории «Администраторы» и используют права, назначенные им администраторами. В графическом интерфейсе WXL-приложений добавлены дополнительные возможности, повышающие удобство работы оператора. Опишем особенности графического интерфейса WXL-приложений, расширяющие спецификацию «*CUA*». Эти расширения касаются всех окон, которые операторы используют для ввода информации. Перечислим их:

- 1) Клавиши перемещения между компонентами ввода-вывода и другими управляющими элементами рабочих окон могут быть настроены под конкретного оператора. Для этих целей могут быть использованы клавиши «*TAB*», «*Shift*» + «*TAB*», «*Enter*» и стрелки. Причём «*TAB*» и «*Shift*» + «*TAB*» используются в любом случае и предназначены для перемещений в двух взаимно обратных направлениях. Вот как выглядит диалог добавления нового пользователя при использовании пункта «Пользователи» меню «Администратор» типового WXL-приложения:

Таким образом, мы видим, что мы можем присваивать стрелкам и клавише *Enter* функцию перемещения между компонентами ввода-вывода и другими управляющими элементами рабочих окон. Следует иметь в виду, что данные клавиши используются независимо друг от друга. Например, мы можем в *WXL*-приложении для перемещений использовать одновременно и *Enter*, и табуляцию, и стрелки. Механизмы воздействия на перемещения между компонентами реализованы в модуле *WxEntry*. Для этих целей созданы глобальные переменные, которые затем изменяются, как только пользователь начинает работать с *WXL*-приложением. Переменная *WxEnterAsTab* присваивается *true*, если клавишу *Enter* необходимо использовать как табуляцию, то есть для перемещений между компонентами. Переменная *WxUpDownAsTab* присваивается *true*, если клавиши «Стрелка вниз» и «Стрелка вверх» необходимо использовать как табуляцию и табуляцию с клавишей *Shift*, то есть для перемещений между компонентами в двух взаимно обратных направлениях.

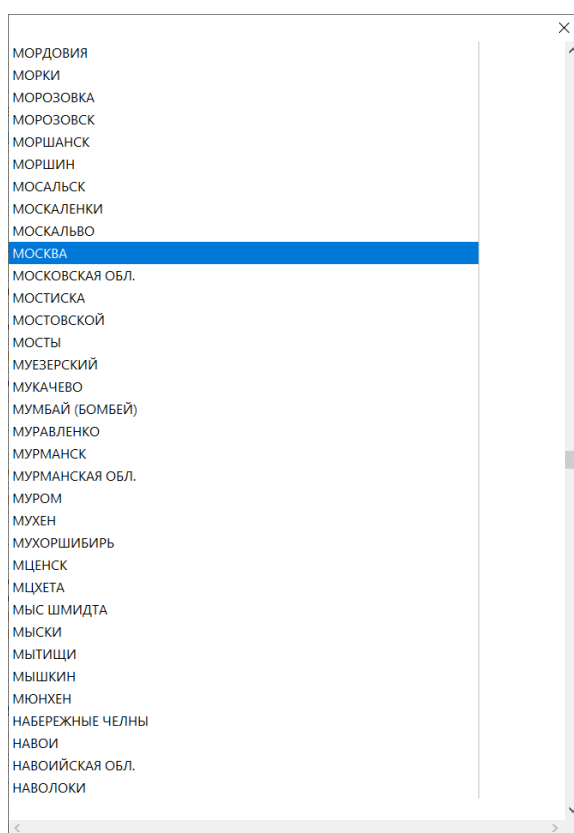
2) К компонентам ввода-вывода библиотеки *WXL* можно присоединять словари (справочники), из которых затем будут извлекаться данные для расположения их на соответствующем компоненте. Присоединение словаря происходит при помощи использования компонента *WxDictButton*. Взаимное расположение компонента ввода-вывода, к которому осуществляется присоединение словаря, и кнопки вызова словаря, определяется свойствами *LinkAttach*, *LinkSeparation* и *LinkControl*, наследуемыми компонентом *WxDictButton* от компонента *WxLinkButton*. Вот как выглядит, например, компонент с присоединённым к нему словарём:  или . Соответственно,  или  – изображение кнопки вызова словаря. Способы вызова присоединённого словаря зависят от настроек рабочего места оператора. Перечислим их:

- Непосредственное нажатие на кнопку левой кнопкой мыши.
- Нажатие на клавишу, соответствующую глобальной переменной *WxDictKey*, определённой в модуле *WxEntry* (по умолчанию «F3»). Для

ЭТОГО КОМПОНЕНТ ВВОДА-ВЫВОДА, К КОТОРОМУ ПРИСОЕДИНЁН СЛОВАРЬ, ДОЛЖЕН НАХОДИТЬСЯ В ФОКУСЕ.

- Нажатие на клавишу, соответствующую глобальной переменной *WxDictKey2*, определённой в модуле *WxEntry* (по умолчанию «Стрелка вниз»). Данный метод работает, только если глобальной переменной *WxUpDownAsTab*, определённой в модуле *WxEntry*, присвоено *false* и компонент ввода-вывода, к которому присоединён словарь, находится в фокусе.

Вот как выглядит, например, форма с данными словарями, вызванная одним из трёх перечисленных выше способов:



Иногда, оператору удобно, чтобы на форме отображались кнопки «ОК» и «Отмена». Присутствие в форме выбора значений из словаря нижней серой панели с кнопками «ОК» и «Отмена» определяется глобальной переменной *WxDictStyle* модуля *WxTblDict*. Необходимо присвоить этой глобальной переменной *dsFormOkCancel*, если необходимо отображать нижнюю серую панель с кнопками, *dsForm*, если отображение этой панели не требуется.

Для подтверждения выбранного значения необходимо нажать клавишу «Enter», левую клавишу мыши или, если она присутствует, кнопку «ОК».

Для перемещений внутри списка используются стрелки. «Стрелка вниз» и «Стрелка вверх» используются для переходов между соседними строками одного столбца. «Стрелка влево» и «Стрелка вправо» используются для переходов между соседними столбцами. Кнопки «PgUp» и «PgDn» используются для переходов на страницу вперёд и назад. Если в словаре только один столбец, то для переходов к началу и концу списка словарных значений используются «Home» и «Ctrl» + «Home», «End» и «Ctrl» + «End». Если в списке словарных значений несколько столбцов, то «Home» и «End» используются для переходов к первому и последнему столбцу внутри строки.

Рассмотрим, что будет занесено на компонент, если мы выбрали значение из всплывающего списка. На это влияет заполнение множества *DictOptions* компонента *WxDictButton*. Если в множество включена опция *dictConcat*, то при выборе нового значения из словаря новое значение будет добавлено через символ, определённый глобальной переменной *WxDictConcatSep* модуля *WxTblDict* (по умолчанию, запятая), к уже имеющемуся содержимому компонента. Если данная опция не включена в список опций кнопки вызова словаря, то значение на компоненте заменяется выбранным из списка. Если несколько колонок, то для определения, значение какого столбца помещать на компонент, необходимо определить свойство *KeyFieldName* источника данных словаря *WxDictSource*.

Если в множестве опций *DictOptions* компонента *WxDictButton* содержится *dictMulti*, то можно выбрать сразу несколько значений из всплывающего списка справочника. Включение нового значения осуществляется нажатием «Insert» на значение, которое должно быть занесено на компонент. Удаления значения из этого списка осуществляется нажатием клавиши «Delete». Сам список отображается в правом верхнем углу формы словаря. При подтверждении выбранного списка стандартным образом на компонент заносятся все значения выбранного списка через символ, определённый глобальной переменной *WxDictConcatSep* модуля *WxTblDict* (по умолчанию, запятая). Причём, если использовалась опция *dictConcat*, то эти значения будут добавлены к существующему содержанию компонента, а если опция не использовалась, то эти значения заменят содержимое компонента.

Отображение заголовков столбцов словаря можно осуществить, добавив в множество опций *DictOptions* компонента *WxDictButton* опцию *dictTitles*. Если заголовки столбцов словаря отображать не требуется, то данная опция не включается.

Если глобальная переменная *WxIncrementalDictSearch* модуля *WxEntry* *true*, а в множестве опций *DictOptions* компонента *WxDictButton* содержится *dictLookUp*, то при ручном вводе значений компонента ввода-вывода будут всплывать подсказки в виде значений из словаря, удовлетворяющих уже введённому данным.

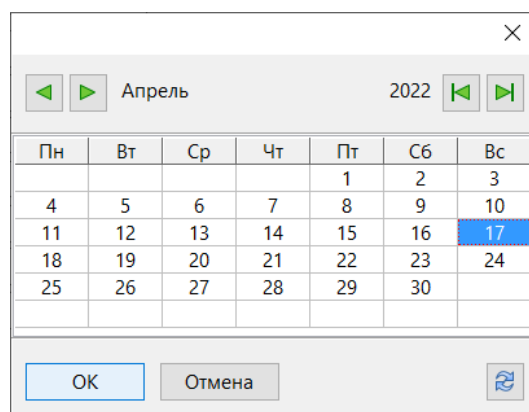
3) Рассмотрим особенности работы с полями ввода дат и времени:

3.1) Если в поле даты введены число и месяц, но не введён год, то проставляется текущий год. Если введено только число, то проставляются текущий месяц и год.

3.2) Допускается введение только двух последних цифр года. Тогда первые две цифры заполняются предыдущим веком, если последние две цифры больше 50, и текущим веком, если последние две цифры меньше или равны 50.

3.3) При нажатии функциональной клавиши «*F4*» на компоненты ввода даты устанавливается текущая дата, на компонент ввода времени устанавливается текущее время.

3.4) При нажатии на клавишу, соответствующую глобальной переменной *WxDictKey*, определённой в модуле *WxEntry* (по умолчанию «*F3*»), появляется календарь, предназначенный для удобного выбора интересующей даты. Вот как выглядит этот календарь:



4) Во многих рабочих окнах программы присутствуют сетки, с помощью которых оператор может просматривать содержимое наборов данных. Рассмотрим возможности, предоставляемые сеткой *WxGrid*.

ID	Номер референса	Действует до	Название компании
▶ 1	550000044	19.07.2024	«Санаторий имени М.В. Фрунзе» ООО «Санаторий име
2	550000105	20.12.2021	Отель "Гала-Альпик" ИП Кравченко Дарья Вячеславов
3	550000228	30.10.2022	Пансионат отдыха "Кавказ" - филиал ООО "Газпром тр
15105	550000237	25.06.2024	«Гостиничный комплекс «Сочи-Магнолия» АО «ГК «Сс
4	550000244	02.12.2021	Отель "Гармония" ИП Узун Елена Ивановна
6	550000300	04.12.2021	Гостиничный комплекс «Бон Апарта» ООО «Бон Апарта»
7	550000301	01.12.2021	Отель Магистрат ООО "Отель Магистрат"
8	550000311	23.12.2022	Гостиница "Айсберг" ООО "Газпром Добыча Надым"
9	550000316	18.08.2022	Мини-отель "Фридрихсхофф" Индивидуальный предп
10	550000317	21.05.2022	Гостиница "Роял Фальке Резорт энд СПА" ("Royal Falke I
11	550000318	26.12.2021	Отель "Балтика"

Нажав на прямоугольник с заголовком столбца левой кнопкой мыши, и, удерживая кнопку мыши в нажатом состоянии, можно менять столбцы местами. В этом случае левая вертикальная граница столбца выделяется красным цветом. При движении мыши влево и вправо красная вертикальная линия последовательно располагается между различными столбцами таблицы. Когда оператор отпускает кнопку мыши, столбец, на котором было осуществлено нажатие, располагается между теми столбцами, которые были разделены красной вертикальной линией. Предыдущее положение столбца удаляется.

Нажав на правую границу прямоугольника с заголовком столбца левой кнопкой мыши, можно изменять видимую ширину столбца. В этом случае курсор изменяет изображение. После нажатия левой кнопкой мыши, удерживая её, необходимо двигать мышь вправо и влево, сдвигая тем самым границу выбранного столбца. Она и будет новой правой границей выбранного столбца.

Вот как, например, может отображаться тот же набор данных после проведённых преобразований:

ID	Название компании	Номер референса	Действует до
▶ 1	«Санаторий имени М.В. Фрунзе» ООО «Санаторий имени М.В. Фрунзе»	550000044	19.07.2024
2	Отель "Гала-Альпик" ИП Кравченко Дарья Вячеславовна	550000105	20.12.2021
3	Пансионат отдыха "Кавказ" - филиал ООО "Газпром трансгаз Нижний Новгород"	550000228	30.10.2022
15105	«Гостиничный комплекс «Сочи-Магнолия» АО «ГК «Сочи-Магнолия»	550000237	25.06.2024
4	Отель "Гармония" ИП Узун Елена Ивановна	550000244	02.12.2021
6	Гостиничный комплекс «Бон Апарта» ООО «Бон Апарта»	550000300	04.12.2021
7	Отель Магистрат ООО "Отель Магистрат"	550000301	01.12.2021
8	Гостиница "Айсберг" ООО "Газпром Добыча Надым"	550000311	23.12.2022
9	Мини-отель "Фридрихсхофф" Индивидуальный предприниматель Вороненко Людм	550000316	18.08.2022
10	Гостиница "Роял Фальке Резорт энд СПА" ("Royal Falke Resort & SPA")	550000317	21.05.2022
11	Отель "Балтика"	550000318	26.12.2021

Нажатие левой кнопки мыши на столбце приводит к сортировке по этому столбцу по возрастанию, если кнопка «*Ctrl*» не удерживается, и по убыванию, если

удерживается. Если удерживать и клавишу «*Shift*», то можно осуществить сортировку по нескольким полям.

Нажимая на правую кнопку мыши в пределах сетки, можно вызвать всплывающее служебное меню сетки.

Каждый пункт меню предназначен для использования дополнительной функциональности сетки *WxGrid*. Рассмотрим вкратце каждую из этих возможностей:

Карточка	F7
Искать поле	Ctrl+F
Искать запись	Ctrl+S
Искать запись дальше	Ctrl+N
Запрос по образцу	
Фильтр строк	
Фильтр колонок	
Шрифт	
Запомнить вид	
Исходный вид	
Экспорт	
Печать	
Информация	

- 1) «Карточка» – данный пункт меню предназначен для просмотра записей в виде карточек. Первая запись, отображаемая на карточке, совпадает с текущей записью на сетке. Далее перемещение между записями набора данных можно осуществлять при помощи присутствующей на карточке системы навигации. Данный пункт меню может быть вызван при помощи горячей клавиши «*F7*».
- 2) «Искать поле» – данный пункт меню предназначен для поиска записи с заданным значением поля, на котором был осуществлён вызов меню правой кнопкой мыши. При этом указатель текущей записи на сетке перемещается к полю с указанным значением, если запись с указанным значением поля найдена. Если запись не найдена, то раздаётся звуковой сигнал и указатель текущей записи на сетке остаётся на прежней позиции. При поиске можно использовать две дополнительные опции: считать заглавные и строчные буквы совпадающими, искать запись, для которой значение поля начинается с указанного значения. Для использования этих возможностей достаточно поставить галочки напротив записей «*a = A*» и «Начинается с ...». Данный пункт меню может быть вызван при помощи сочетания клавиш «*Ctrl*» + «*F*».

Искать поле

Номер референса

55566

a = A Начинается с ...

OK Отмена

- 3) «Искать запись» – данный пункт меню предназначен для поиска записи с заданными условиями на значения полей. Если поиск удачный, то указатель текущей записи на сетке перемещается к записи, удовлетворяющей заданным условиям. Если поиск не удачный, то возникает сообщение «Запись не найдена». Правила задания условий на значения столбцов подчиняются реляционному исчислению доменов. По умолчанию, поиск осуществляется только среди записей, расположенных ниже указателя текущей записи на сетке. Ставя крестик напротив «Вверх», можно учитывать только записи, расположенные выше указателя текущей записи на сетке. Ставя крестик напротив «Глобальный поиск», можно осуществлять поиск записей на всём отображаемом наборе данных. Ставя крестик напротив «a = A», можно не учитывать при поиске регистр букв. Данный пункт меню может быть вызван при помощи сочетания клавиш «**Ctrl**» + «**S**».

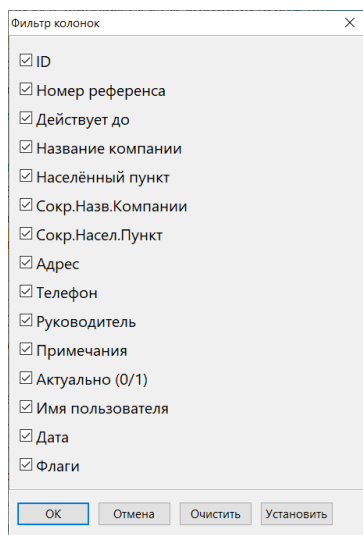
ID	
Название компании	
Номер референса	
Действует до	
Населённый пункт	
Сокр.Назв.Компании	
Сокр.Насел.Пункт	
Адрес	
Телефон	
Руководитель	
Примечания (МЕМО)	
Актуально (0/1)	
Имя пользователя	
Дата	
Флаги	

OK Отмена Очистить a = A Глобальный поиск Вверх

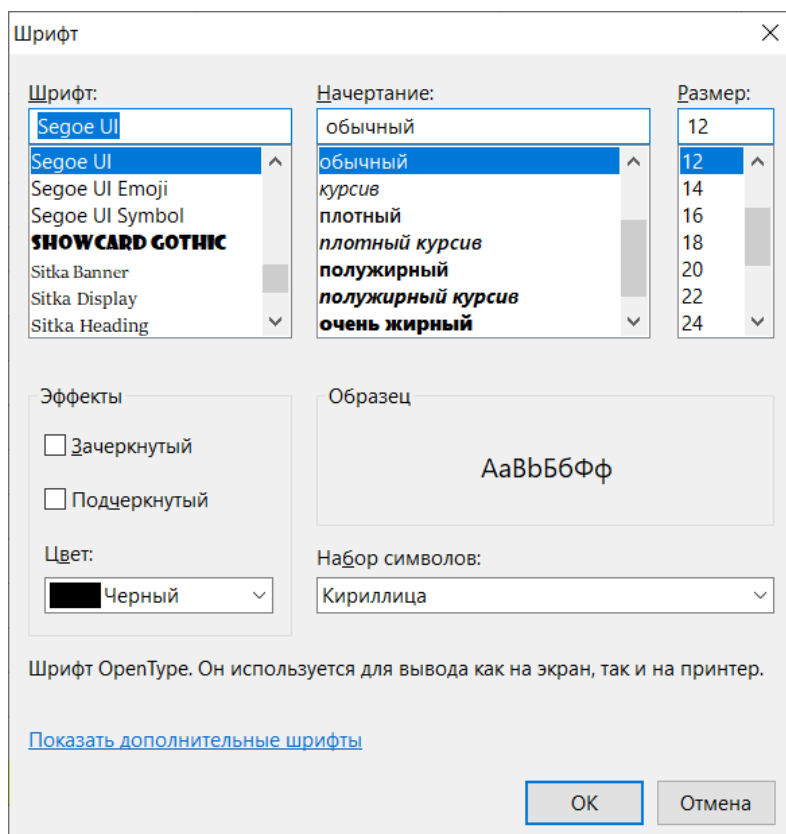
- 4) «Искать запись дальше» – данный пункт меню предназначен для поиска следующей записи, удовлетворяющей набору условий, указанному в предыдущем пункте. Данный пункт меню может быть вызван при помощи сочетания клавиш «**Ctrl**» + «**N**».
- 5) «Запрос по образцу» – данный пункт меню предназначен для отображения на отдельном окне в отдельной сетке всех записей, удовлетворяющих условиям, заданным в соответствии с шаблоном. Пример шаблона: `[CityName]='M%'`.

Этот шаблон отобразит все записи, в которых населённый пункт начинается на букву «М».

- 6) «Фильтр строк» – данный пункт меню предназначен для отображения на текущем окне в текущей сетке всех записей, удовлетворяющих условиям, заданным в соответствии с шаблоном.
- 7) «Фильтр колонок» – данный пункт меню предназначен для выбора, какие столбцы требуется отображать.

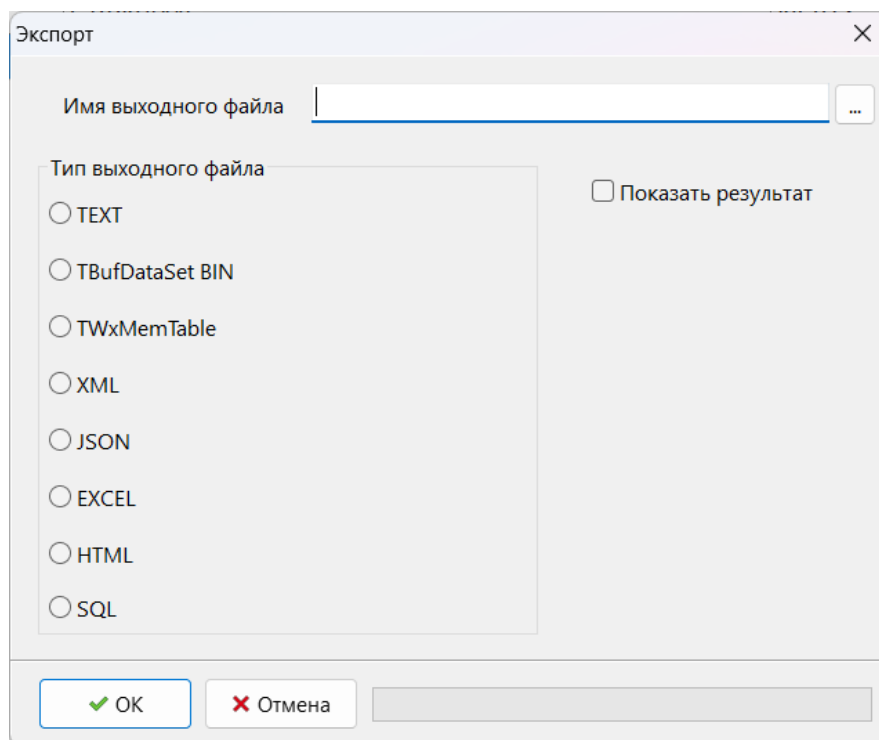



- 8) «Шрифт» – данный пункт меню предназначен для выбора шрифта, каким отображаются строки отображаемой таблицы.



- 9) «Запомнить вид» – данный пункт меню предназначен для запоминания пользовательских настроек сетки. В типовом WXL-приложении вид сетки запоминается рядом с исполняемым файлом приложения в файле, который имеет вид ИМЯТАБЛИЦЫ_СУБД_ТЕХНОЛОГИЯ.lay (например, *turhotels_mssql_zeos.lay*). Запоминаемые настройки включают порядок столбцов и их ширину, шрифт, используемый для отображения строк набора данных.
- 10) «Исходный вид» – данный пункт меню предназначен для возвращения к принятым по умолчанию настройкам сетки.
- 11) «Экспорт» – данный пункт меню предназначен для экспорта набора данных в один из следующих форматов:
- 11.1) Текстовый файл.
 - 11.2) *TBufDataSet BIN* (при наличии *{ \$DEFINE USEBUFDATASET }*).
 - 11.3) *TWxMemTable*.
 - 11.4) *XML* (при наличии *{ \$DEFINE EXPORTXML }*).
 - 11.5) *JSON* (при наличии *{ \$DEFINE EXPORTJSON }*).
 - 11.6) *EXCEL*.
 - 11.7) *HTML*.
 - 11.8) *SQL*.
 - 11.9) *DBASE* (при наличии *{ \$DEFINE USETDBF }*).
 - 11.10) *TkbmMemTable BIN* (при наличии *{ \$DEFINE USEKBMOMEMTABLE }*).
 - 11.11) *TkbmMemTable CSV* (при наличии *{ \$DEFINE USEKBMOMEMTABLE }*).
 - 11.12) *TVirtualTable BIN* (при наличии *{ \$DEFINE USEVIRTUALTABLE }*).
 - 11.13) *TVirtualTable XML* (при наличии *{ \$DEFINE USEVIRTUALTABLE }*).

Выбор форматов определяется настройками в файле «*wxdefs.inc*». Если указанный в скобках *DEFINE* присутствует, то пункт меню отображается. Если указанный в скобках *DEFINE* отсутствует, то пункт меню не отображается. Если примечание в скобках не указано, то пункт меню отображается всегда. Один из возможных вариантов формы показан на рисунке:



- 12) «Печать» – данный пункт меню предназначен для вывода содержимого таблицы на принтер. Предварительный просмотр (*Preview*) строится с использованием возможностей выбранного генератора отчётов (*FastReport* или *LazReport*).
- 13) «Информация» – данный пункт меню предназначен для получения информации о столбцах таблицы, файле настроек отображения, числе записей в отображаемом наборе данных и их размере.
- 5) Во многих окнах WXL-приложений, предназначенных для ввода информации, присутствует кнопка , которая предназначена для подтверждения оператором факта завершения оператором ввода информации в текущем окне.

6. Доступ к данным

6.1 Технологии доступа к данным из приложения

Доступ к локальным базам данных и понятие одноуровневой архитектуры

Предположим, что работа осуществляется с базой данных, находящейся на том же компьютере, что и программа для её использования. Кроме того, вся обработка данных осуществляется непосредственно в приложении. Соответственно, доступ к таблицам всегда осуществляется со стороны только одного клиента. Такая схема применяется для работы с небольшими объёмами информации и представляет собой одноуровневую архитектуру для работы с базами данных. Приложения такого вида обычно работают с локальными таблицами *DBASE*, *PARADOX*, *FOXPRO*, *MSACCESS*, *SQLITE* или просто с текстовыми файлами. Для подобного рода приложений в библиотеке *WXL* реализован модуль *WxDBISAM*, через который осуществляется выбор методов работы как с базами данных к информации, упорядоченной с помощью инсталлируемого индексно-последовательного метода доступа *ISAM* (*Indexed Sequential Access Method*). К таким данным относятся ранее перечисленные локальные таблицы, данные, хранимые в рабочих книгах *Excel*, почтовые файлы *Outlook* и *Microsoft Exchange*, текстовые и *HTML* файлы. В настоящее время из всего перечисленного набора в библиотеке *WXL* поддерживается работа только с таблицами *DBASE* при помощи компонента *TDBF*.

TDBF представляет собой бесплатный компонент прямого доступа для *Delphi* и *Lazarus*. Созданные при помощи данного средства программы для работы с базами данных компактны и не требуют предварительной инсталляции. Подходит для работы с таблицами *DBASE*, *Clipper* и *Visual FoxPro*.

Двухуровневая архитектура и СУБД типа клиент-сервер

Во многих системах управления базами данных существуют библиотеки, содержащие специальный интерфейс прикладного программирования (*application programming interface – API*), который представляет собой набор функций для манипулирования данными. В СУБД типа клиент-сервер *API* (в данном случае, он

также именуется клиентским программным обеспечением систем управления базами данных) инициирует отправку по сети запроса к серверу и получение результатов или кодов ошибок для дальнейшей их обработки клиентским приложением. Подобный способ организации работы с базами данных представляет собой двухуровневую архитектуру.

Наиболее распространённый способ доступа к данным заключается в непосредственном использовании *API*. Однако это означает полную зависимость приложения от используемой системы управления базами данных. В этом случае переход к другой системе влечёт за собой переписывание большей части программного кода клиентского приложения. Отсюда мы приходим к мысли о необходимости универсального механизма доступа к данным, обеспечивающего для клиентского приложения стандартный набор общих функций, классов или сервисов (служб), необходимых для работы с различными системами управления базами данных. Эти стандартные функции (классы или сервисы) должны размещаться в библиотеках, именуемых драйверами или провайдерами баз данных (*data base drivers (providers)*). Каждая такая библиотека реализует набор стандартных функций, классов или сервисов, используя обращения *API* к конкретной системе управления базами данных.

Рассмотрим наиболее популярные механизмы доступа к данным (*Universal Data Access, UDA*), работа с которыми поддерживается в библиотеке *WXL*.

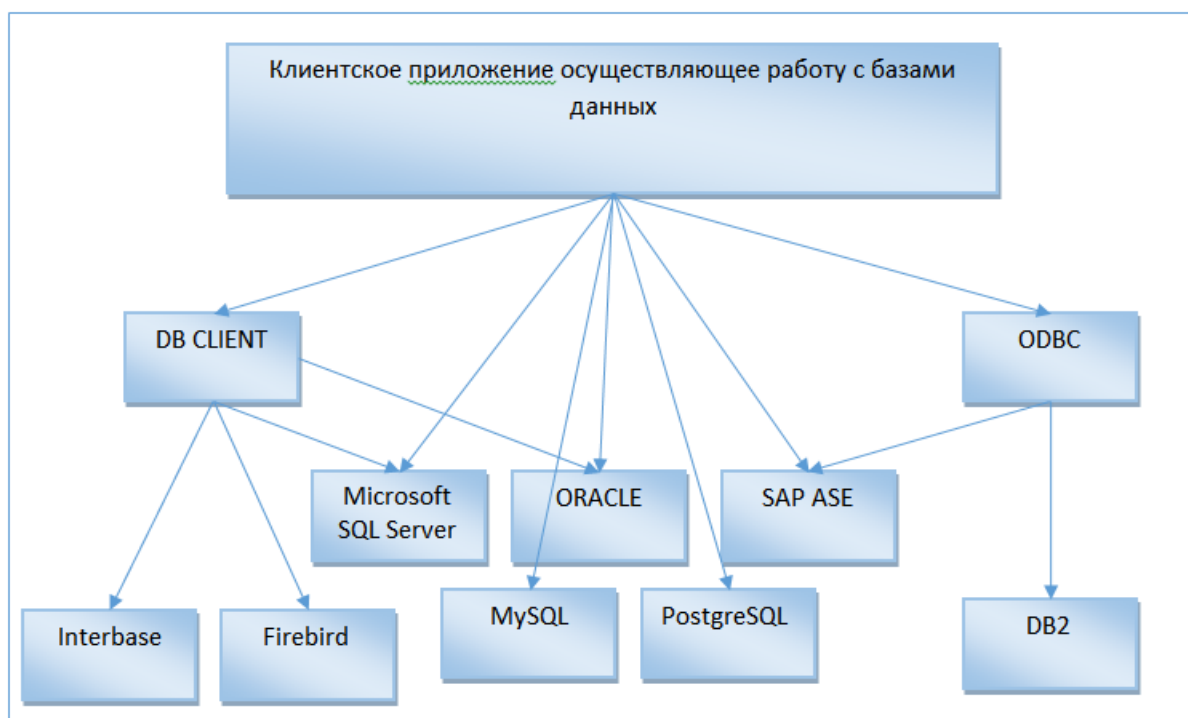
ODBC (Open Database Connectivity – открытые средства связи с базами данных) представляют собой широко применяемый пользовательский интерфейс, отвечающий требованиям стандартов *ANSI* и *ISO*, предъявляемых к интерфейсу на уровне обращений к базам данных (*database call level interface (CLI)*). Для доступа к реляционной базе данных посредством *ODBC* необходимо, чтобы был установлен так называемый администратор *ODBC (ODBC Administrator)* и соответствующий драйвер *ODBC*. Драйвер *ODBC* представляет собой динамически подключаемую библиотеку (*dynamic link library – DLL*), которую клиентское приложение может использовать для доступа к источнику данных *ODBC*. В *ODBC* существуют драйверы для каждой поддерживаемой СУБД, так как в них используются вызовы *API* клиентского программного обеспечения этой СУБД. *ODBC* обеспечивает

поддержку доступа к любым базам данных (и даже к некоторым другим файлам, например к электронным таблицам), для которых имеется соответствующий драйвер *ODBC*.

SQLdb – это свободная (лицензии *GPL*, *LGPL*) и открытая библиотека компонентов для доступа к данным, которая поддерживается на различных операционных системах и разрабатывается сообществом разработчиков *Free Pascal* + *Lazarus*. Технология *SQLdb* используется в среде программирования *Lazarus* для доступа к базам данных по умолчанию, то есть она входит в «коробочный» вариант поставки среды *Lazarus*. Начало разработки *SQLdb* датируется 2004 годом. В *SQLdb* поддерживается работа с *Microsoft SQL Server*, *Sybase ASE*, *PostgreSQL*, *ORACLE*, *MySQL*, *SQLite*, *Interbase* и *Firebird*. Кроме того, для работы с иными СУБД возможно использование технологии *ODBC* через базовый компонент *TSQLConnector* и соответствующие драйверы *ODBC*.

UniDAC (*Universal Data Access Components*) – это универсальная проприетарная технология доступа к данным от компании *Devart*, которая поддерживается на различных операционных системах (*Windows*, *Mac OS X*, *iOS*, *Android*, *Linux* и *FreeBSD*) для 32-битных и 64-битных приложений. Компания *Devart* была основана в 1997 году и с самого своего основания специализировалась на технологиях доступа к данным. К настоящему моменту её продуктами пользуются более 40,000 пользователей, а сама компания является партнёром *Microsoft* (*Silver Application Development Partner*) и *ORACLE* (*Silver Partner in the Oracle Partner Network (OPN)*). Первая версия продукта *UniDAC* вышла 23 апреля 2008 года. То есть сама компания существует уже более 20 лет, а её продукт *UniDAC* более 10 лет. Компоненты доступа к данным на основе технологии *UniDAC* не являются встроенными ни в среду программирования *Delphi*, ни в среду программирования *Lazarus*, но их возможно купить и установить в обеих средах (в том числе, с исходным кодом). При использовании технологии *UniDAC* возможно работать с различными источниками данных напрямую, без использования клиентских библиотек (*Microsoft SQL Server*, *ORACLE*, *MySQL*, *PostgreSQL*, *SQLite*, *NexusDB*), через использование *DB Client* и/или *ODBC*. В качестве независимых компонентов компания *Devart* разработала компоненты для доступа к отдельным

СУБД: *ODAC* (*Oracle Data Access Components*), *SDAC* (*SQL Server Data Access Components*), *MyDAC* (*Data Access Components for MySQL*), *IBDAC* (*InterBase Data Access Components*), *PgDAC* (*PostgreSQL Data Access Components*), *LiteDAC* (*SQLite Data Access Components*).



ТЕХНОЛОГИЯ ДОСТУПА К ДАННЫМ *UNIDAC*

ZeosDBO – это свободная (лицензии *GPL*, *LGPL*) и открытая библиотека компонентов для доступа к данным, которая поддерживается на различных операционных системах (*Windows*, *Linux* и *FreeBSD*) и разрабатывается сообществом разработчиков *ZeosLib*. Начало разработки *ZeosLib* датируется 1999 годом. В настоящее время поддерживаются следующие источники данных: *MySQL*, *PostgreSQL*, *Interbase*, *Firebird*, *Microsoft SQL Server*, *Sybase ASE*, *ORACLE*, *SQLite*. Компоненты доступа к данным на основе библиотеки *ZeosDBO* не являются встроенными ни в среду программирования *Delphi*, ни в среду программирования *Lazarus*, но их возможно скачать и установить в обеих средах. Начиная с версии 7.3.x, в библиотеке *ZeosLib* реализована поддержка *ADO*, *OLE DB* и, что особенно важно, – *ODBC*.

OLE DB (возможно использование посредством *ZeosDBO*) обеспечивает низкоуровневый интерфейс для доступа к данным. *OLE DB* служит для обеспечения доступа ко всем типам данных с использованием компонентной

объектной модели (*Component Object Model, COM*) посредством *COM*-интерфейсов. Для доступа к конкретным источникам данных с помощью *OLE DB* необходимо установить соответствующий провайдер *OLE DB (OLE DB Provider)*. Он представляет собой динамически подключаемую библиотеку (*DLL*), которую клиентское приложение может использовать для получения доступа к некоторому источнику данных *OLE DB*. Каждый провайдер *OLE DB* является специфическим для той или иной системы управления базами данных, так как он использует вызовы *API* клиентского программного обеспечения этой конкретной СУБД.

Microsoft ActiveX Data Objects (ADO) (возможно использование посредством *ZeosDBO*) представляет собой интерфейс прикладного программирования для доступа ко всем источникам данных, в том числе нереляционным. Существуют расширения *ADO*, такие как *ADOX (ADO extensions for DDL and security)*, *JRO (Jet Replications Objects)* и *ADO MD (ADO Multidimensional)*. Технология доступа к данным *ADO* использует *OLE DB*. *ADO* содержит набор библиотек, реализующих объектную модель *ADO*, которая может применяться в клиентских приложениях. В настоящее время *ADO* является непосредственной частью операционных систем семейства *Windows*.

Компоненты для доступа к данным: компонент для работы с запросом, компонент для работы с транзакцией, компонент доступа к серверной базе данных

Для работы с базами данных в среде программирования *Lazarus* используются специально написанные для этих целей компоненты. Наиболее часто используемыми являются невидимые компоненты для работы с таблицей, *SQL*-запросом, соединением с базой данных и транзакцией. Все возможности компонента для работы с таблицей можно реализовать при помощи компонента для работы с *SQL*-запросом, поэтому такой компонент отсутствует в библиотеке *WXL*. В библиотеке *WXL* осуществлено создание единого интерфейса вне зависимости от выбранной технологии доступа к данным. Пользователь определяет предпочтительную для использования в своих приложениях технологию доступа к реляционным данным *SQL*-сервера с помощью директив условной компиляции,

указанных в файле «*wxdefs.inc*». После этого ему уже не приходится заботиться о конкретных особенностях использования той или иной технологии доступа к данным. Библиотека *WXL* позволяет использовать *SQLDb*, *UniDAC*, *ZeosDBO* для доступа к данным в архитектуре типа клиент-сервер. Занесём перечисленные компоненты в таблицу для всех поддерживаемых библиотекой *WXL* технологий доступа к данным.

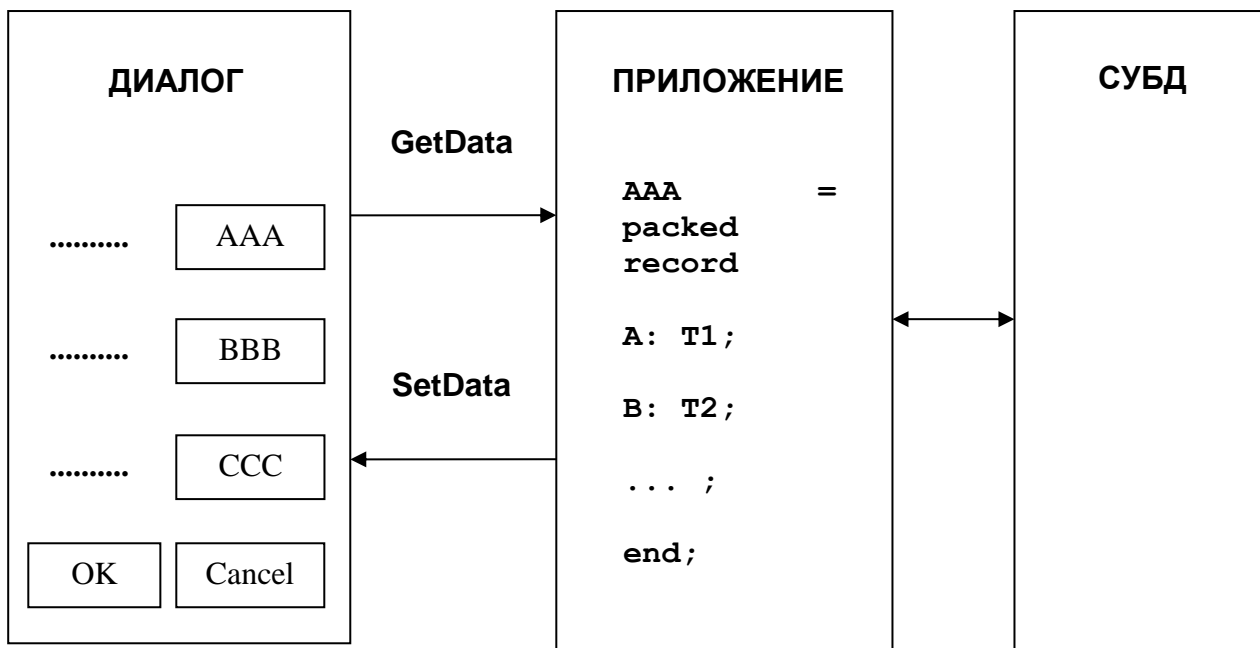
	Запрос	Транзакция	Доступ к данным
<i>SQLDb</i>	<i>TSQLQuery</i>	<i>TSQLTransaction</i>	<i>TSQLConnector</i>
<i>UniDAC</i>	<i>TUniQuery</i>	<i>TUniTransaction</i>	<i>TUniConnection</i>
<i>ZeosDBO</i>	<i>TZQuery</i>	<i>TZTransaction</i> , если <i>UseTZTransaction</i> <i>TZConnection</i> , если нет	<i>TZConnection</i>
<i>WXL</i>	<i>TWxSQLQuery</i>	<i>TWxSQLTransaction</i>	<i>TWxSQLConnection</i>

6.2 Проблема пустых данных (*NULL*)

В языках программирования переменные относятся к определённым типам. В таблицах баз данных есть поля, которые также относятся к своим серверным типам, но могут принимать значение *NULL*. Каждой таблице рассматриваемой базы данных в приложении ставится в соответствие запись, состоящая из полей, отнесённых к определённым типам данных используемого языка программирования. Для отображения этих записей пользователю используется набор компонентов ввода-вывода. В библиотеке *WXL* – это типизированные компоненты ввода-вывода, наследуемые от *TWxEntry* и *TWxMemo*. Компоненты ввода-вывода также могут быть не заполнены. Для упрощения процесса передачи данных между расположенными на форме компонентами ввода-вывода данных и записями приложения, соответствующими записям таблицы баз данных, используется компонент *TWxTransfer*. В результате для занесения записи на форму достаточно вызвать процедуру *SetData* компонента *TWxTransfer*, а для заполнения записи данными с формы достаточно вызвать процедуру *GetData* компонента *TWxTransfer*. Параметром для данных процедур является переменная типа записи, соответствующей записи таблицы базы данных.

При каждой передаче данных: между формой и приложением и между приложением и сервером баз данных – возникает вопрос, как воспринимать пустые

значения, допустимые компонентами формы и сервером баз данных, но недопустимые в самом приложении баз данных.



Легче всего данная проблема решается, если поле таблицы баз данных не может содержать нулевые значения. В этом случае компонент ввода-вывода устанавливается на недопустимость пустых значений и, если данные на такой компонент не введены, возникает ошибка при проверке правильности заполнения компонента. Установить компонент ввода-вывода библиотеки *WXL* на недопустимость пустых значений можно присвоением соответствующего свойства *BlankEnable*. Проверить правильность заполнения любого компонента ввода-вывода библиотеки *WXL* можно при помощи процедуры *ValidateEntry*. Если свойство *SoftValidation* компонента ввода-вывода установлено на *false*, то проверка правильности содержимого компонента ввода-вывода осуществляется каждый раз, когда компонент ввода-вывода теряет фокус.

Однако возникает вопрос, что же делать, если поле таблицы базы данных может содержать пустое значение. В библиотеке *WXL* предлагается следующее решение данной проблемы: для разных типов данных выбираются принадлежащие этим типам данных значения, которые будут восприниматься как пустые при обмене информацией с компонентами ввода-вывода и сервером баз данных. Функциональность для проверки и установки пустых значений реализована в

модуле *WxBlanks*. Занесем информацию о значениях, которые рассматривают в качестве пустых, в таблицу.

Тип данных Lazarus	Значение, рассматриваемое как пустое.
<i>string</i>	"
<i>ShortString</i>	"
<i>AnsiString</i>	"
<i>WideString</i>	"
<i>UnicodeString</i>	"
<i>Byte</i>	0 или <i>High(Byte)</i>
<i>Word</i>	0 или <i>High(Word)</i>
<i>LongWord</i>	0 или <i>High(LongWord)</i>
<i>ShortInt</i>	0 или <i>Low(ShortInt)</i>
<i>Smallint</i>	0 или <i>Low(smallint)</i>
<i>Integer</i>	0 или <i>Low(integer)</i>
<i>Largeint = int64</i>	0 или <i>Low(largeint)</i>
<i>Single</i>	0.0 или -3.4e+38
<i>Double</i>	0.0 или -1.7e+308
<i>Extended</i>	0.0 или -1.1e+4932
<i>Currency</i>	0.0 или -922337203685477 = <i>SYSUTILS.MinCurrency</i>
<i>TBCD</i>	<i>NullBCD</i> или <i>xBlankTBcd</i>
	<i>xBlankBoolean</i>
<i>TDateTime</i>	2958465.999999999, соответствует 31.12.9999 23:59:59

Таким образом, мы видим, что для строк пустым значением считается пустая строка ". Для целых и вещественных чисел определена логическая (*boolean*) переменная *WxBlankAsZero*. Если она истинна (*true*), то в качестве пустого значения используется 0 для целых чисел и 0.0 для вещественных чисел. Если она ложна, то в качестве пустого значения используется наибольшее число диапазона для беззнаковых целых чисел и наименьшее число диапазона для знаковых целых чисел. Для вещественных чисел, если *WxBlankAsZero* ложно, константы заданы непосредственно в модуле *WxBlanks* и приведены в таблице выше. В качестве пустой даты принята переменная типа *TDateTime*, соответствующая дате «31.12.9999 23:59:59». Посредством директивы *overload* реализована функция *IsBlank*, проверяющая соответствие значения принятому по умолчанию пустому значению, и функция *SetBlank*, устанавливающая пустое значение в заданную переменную. Их можно использовать для любых типов данных.

В то же время следует указать, что проверки на соответствие выбранным пустым значениям реализованы в методах ввода и вывода данных компонентов ввода-вывода библиотеки *WXL*. Если при вызове *SetData* требуется установить

значение, соответствующее принятому по соглашению пустому значению, то компонент остаётся незаполненным. В то же время, если компонент не заполнен и это допустимо, то при вызове *GetData* в переменную заносится значение, соответствующее пустому значению для выбранного типа данных.

6.3 Наборы данных в памяти и компонент *TWxMemDataSet*

Изображение на палитре компонентов: .

Изображение на форме при её дизайне: .

Изображение на форме во время исполнения программы: Невизуальный компонент.

Назначение: Набор данных в виде таблицы в памяти, который не связан с какой-либо базой данных. Данный компонент наследуется от класса *TDataSet*, что позволяет использовать все его методы и свойства. Используется в качестве экземпляра временной таблицы, который можно создать в памяти.

Пример использования:

```
{Добавить в uses WxMemDataSet}

{Объявление}
type
  TTestMemDataSet = class(TWxMemDataSet)
  protected
    procedure InternalInitFieldDefs(); override;
    procedure OpenAndFill();
  end;

{Реализация}

procedure TTestMemDataSet.InternalInitFieldDefs();
begin
  FieldDefs.Clear();
  FieldDefs.Add('fCode', ftString, 4, False);
  FieldDefs.Add('fValue', ftInteger, 0, False);
end;

procedure TTestMemDataSet.OpenAndFill();
begin
  Open();
  AppendRecord(['A001', 1]);
  AppendRecord(['A002', 2]);
end;

{Пример использования}

procedure UseTestMemDataSet();
```

```

var
  MDS: TTestMemDataSet;
begin
  MDS := TTestMemDataSet.Create(nil);
  try
    MDS.OpenAndFill();
    MDS.First();
    while not MDS.EOF do
    begin
      {Здесь можно выполнить любые действия с MDS в нуждах приложения}
      {Например, выполнить последовательное отображение кодов в виде сообщений}
      ShowMessage(MDS.FieldName('fCode').AsString);
      MDS.Next();
    end;
  finally
    MDS.Free();
  end;
end;

```

6.4 Модули доступа к данным библиотеки wxl

- 1) Специфичные для технологии доступа к данным *SQLdb* (размещены в подпапке *sqldb* исходников библиотеки *WXL*).
- 1.1) Модуль *WxDbDecimal* содержит функцию *ScaledInt64ToBcd* для преобразования *Int64* в *BCD* с указанием числа знаков после запятой (масштаб, *scale*).
- 1.2) Модуль *WxDbSQL_FPC* обеспечивает доступ к *SQL*-серверам с использованием технологии доступа к данным *SQLDb*.
- 1.3) Модуль *WxDbSQL_FPCUtils* содержит реализацию режима отладки для выполняемых *SQL*-запросов с использованием технологии доступа к данным *SQLDb*.
- 1.4) Модуль *WxODBC* является вспомогательным для модуля *WxODBCConn*.
- 1.5) Модуль *WxODBCConn* является вспомогательным для модуля *WxDbSQL_FPC* для обеспечения работы с *SQL*-серверами через *ODBC* с использованием технологии доступа к данным *SQLDb*.
- 1.6) Модуль *WxODBCSQL* является вспомогательным для модулей *WxODBC* и *WxODBCConn*.
- 2) Специфичные для технологии доступа к данным *UniDAC* (размещены в подпапке *unidac* исходников библиотеки *WXL*).

- 2.1) Модуль *WxDbSQL_UNI* обеспечивает доступ к *SQL*-серверам с использованием технологии доступа к данным *UNIDAC*.
- 2.2) Модуль *WxDbSQL_UNIUtils* содержит реализацию режима отладки для выполняемых *SQL*-запросов с использованием технологии доступа к данным *UNIDAC*.
- 2.3) Модуль *WxUniMapRuleDlg* – диалог для сопоставления поля типу данных.
- 2.4) Модуль *WxUniMapRuleFra* – фрейм для сопоставления поля типу данных.
- 2.5) Модуль *WxUniMapRulesDlg* – диалог для сопоставления полей и типов данных.
- 2.6) Модуль *WxUniMapRulesFra* – фрейм для сопоставления полей и типов данных.
- 2.7) Модуль *WxUniSpecificOptionsDlg* – диалог для задания расширенных настроек соединения *UniDAC* (*SpecificOptions*).
- 2.8) Модуль *WxUniSpecificOptionsFra* – фрейм для задания расширенных настроек соединения *UniDAC* (*SpecificOptions*).
- 2.9) Модуль *WxUniTypes* – вспомогательный модуль для модулей сопоставления полей и типов данных: *WxUniMapRuleDlg*, *WxUniMapRuleFra*, *WxUniMapRulesDlg* и *WxUniMapRulesFra*.
- 3) Специфичные для технологии доступа к данным *ZeosDBO* (размещены в подпапке *zeos* исходников библиотеки *WXL*).
 - 3.1) Модуль *WxDbSQL_ZEOS* обеспечивает доступ к *SQL*-серверам с использованием технологии доступа к данным *ZeosDBO*.
 - 3.2) Модуль *WxDbSQL_ZeosUtils* содержит реализацию режима отладки для выполняемых *SQL*-запросов с использованием технологии доступа к данным *ZeosDBO*.
- 4) Прочие *GUI*-модули для доступа к данным (размещены в подпапке *gui* исходников библиотеки *WXL*).
 - 4.1) Модуль *WxAsyncQuery* содержит реализацию класса для асинхронного выполнения *SQL*-запроса.
 - 4.2) Модуль *WxDbSQLConnectDlg* содержит реализацию диалога для соединения с СУБД.
 - 4.3) Модуль *WxDbSQLConnectFrm* содержит реализацию формы для соединения с выбранной СУБД (в частности, используется в генераторе *wxlgen*).

- 4.4) Модуль *WxDbSQLConnectList* содержит реализацию набора данных для хранения строк соединения на основе *TWxMemDataSet*.
- 4.5) Модуль *WxDbSQLConnectListFrm* содержит форму для выполнения выбранных действий над набором данных для хранения строк соединения: добавление соединения, исправление соединения, удаление соединения, проверка соединения.
- 4.6) Модуль *WxDbSQLConnectPanel* содержит реализацию вспомогательного класса *TWxDbSQLConnectFrame* для модуля *WxDbSQLConnectDlg* на основе панели (*TPanel*).
- 5) Прочие модули без *GUI* для доступа к данным.
- 5.1) Модуль *WxConnectString* содержит функциональность по построению строки соединения.
- 5.2) Модуль *WxDbISAM* обеспечивает функциональность для работы с локальными базами данных.
- 5.3) Модуль *WxDbParams* предназначен для поддержки работы с параметрами *TParam*.
- 5.4) Модуль *WxDbSQL* обеспечивает доступ к *SQL*-серверам. Использует функциональность одного из модулей *WxDbSQL_FPC*, *WxDbSQL_UNI* или *WxDbSQL_Zeos* в зависимости от выбранных директив условной компиляции.
- 5.5) Модуль *WxDbSQLProху* является вспомогательным для модуля *WxDbSQLConnectPanel*.
- 5.6) Модуль *WxMemDataSet* содержит реализацию набора данных в виде таблицы в памяти *TWxMemDataSet*, который не связан с какой-либо базой данных.

7. Манипулирование данными

7.1 Компоненты ввода

В библиотеке *WXL* были созданы типизированные компоненты для ввода-вывода типизированных данных. Кроме того, все компоненты ввода-вывода имеют схожий интерфейс, что позволило при помощи компонента *TWxTransfer* связывать их в единую структуру. В результате удаётся вызовом одной функции как считать данные со всех компонентов, объединённых *TWxTransfer*, так и заполнить все компоненты данными записи одновременно. Среди общих возможностей всех компонентов ввода-вывода следует указать и возможность подключения словарей, которые позволяют заполнять компоненты значениями из заранее заданного списка значений, что значительно снижает процент ошибок пользователей при наборе данных. Более подробную информацию, касающуюся работы с компонентами ввода-вывода, можно получить из главы 5, посвящённой основам графического интерфейса *wxl*-приложений.

7.1.1 Однострочные компоненты для ввода строк

Название	Изображение на палитре компонентов	Изображение на форме при её дизайне	Изображение на форме во время исполнения программы	Назначение
<i>TWxEntryShortString</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод строковых данных типа ShortString .
<i>TWxEntryAnsiString</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод строковых данных типа AnsiString .
<i>TWxEntryWideString</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод строковых данных типа WideString .
<i>TWxEntryUnicodeString</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод строковых данных типа UnicodeString .
<i>TWxEntryString</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод строковых данных типа string .

Пример использования 1:

- 1) Помещаем компонент на форму.
- 2) Делаем присвоение `WxEntryString1.MaxLength := 3;`
- 3) В результате ввести строчки, содержащие больше, чем три символа, не удастся.

Пример использования 2:

- 1) Помещаем компонент на форму.
- 2) Помещаем на форму компонент ввода-вывода типа TEdit Edit1.
- 3) Записываем на Edit1 0,1 или оставляем его содержимое без изменений.
- 4) Выполняем следующую процедуру:





```
procedure TForm1.Button1Click(Sender: TObject);
begin
  WxEntryStr1.Text := '';
  if (Edit1.Text='1') then WxEntryStr1.CharCase := ecUpperCase
  else if (Edit1.Text='2') then WxEntryStr1.CharCase := ecLowerCase
  else WxEntryStr1.CharCase := ecNormal;
end;
```



- 5) В результате, если Edit1 содержит 1, то разрешено вводить только заглавные буквы. Если Edit1 содержит 2, то разрешено вводить только строчные буквы. В противном случае, разрешен ввод и тех, и других.

Пример использования 3:

- 1) Помещаем компонент на форму.
- 2) Делаем присвоение `WxEntryWideString1.PasswordChar := '*'`;
- 3) В результате любые введенные символы будут отображаться на экране как *. Данное свойство используется при введении информации, которая не должна быть узнаваема видящими монитор окружающими людьми (например, при наборе паролей).

7.1.2 Компоненты для ввода целых чисел

Название	Изображение на палитре компонентов	Изображение на форме при её дизайне	Изображение на форме во время исполнения программы	Назначение
<i>TWxEntryInteger</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод целых чисел типа integer .
<i>TWxEntryLargeInt</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод целых чисел типа LargeInt .
<i>TWxEntrySmallInt</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод целых чисел типа SmallInt .
<i>TWxEntryShortInt</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод целых чисел типа ShortInt .
<i>TWxEntryWord</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод целых чисел типа Word .

<i>TWxEntryLongWord</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод целых чисел типа LongWord .
<i>TWxEntryByte</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод целых чисел типа Byte .

Пример использования 1:

- 1) Помещаем компонент на форму.
- 2) Вводим на компонент целое число.
- 3) Добавляем компонент *Edit1* класса *TEdit*.
- 4) Заполняем из программы значение *Edit1* значением *WxEntryInteger1*.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  str: integer;
begin
  WxEntryInteger1.GetDataPrim(str);
  Edit1.Text := IntToStr(str);
end;
```

В результате на компоненте *Edit1* отобразится значение, введённое на компонент *WxEntryInteger1*.

Пример использования 2:

- 1) Помещаем компонент на форму.
- 2) Программно записываем на компонент целое число -32768 и проверяем, является ли данное число так называемым «пустым значением».

```
procedure TForm1.Button1Click(Sender: TObject);
var
  MySmallInt: smallint;
begin
  MySmallInt := -32768;
  WxEntrySmallInt1.SetDataPrim(MySmallInt);
  if WxEntrySmallInt1.IsBlankText(WxEntrySmallInt1.Text) then ShowMessage('BLANK');
end;
```

В результате появится сообщение «*BLANK*», если константа *WxBlankAsZero* модуля *WxBlanks* *false*. Если бы использовалось присвоение переменной *MySmallInt* 0, то сообщение появлялось бы при значении *WxBlankAsZero* *true*. В любом случае, на компоненте отображается "", если введённое число соответствует пустому значению.

Пример использования 3:

- 1) Помещаем компонент на форму.

- 2) Вводим на компонент целое число.
- 3) Выводим сообщение со значением *WxEntryLargeInt1*.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  str: integer;
begin
  WxEntryInteger1.GetDataPrim(str);
  ShowMessage(IntToStr(str));
end;
```

В результате появится сообщение со значением, введённым на компонент *WxEntryLargeInt1*.

Пример использования 4:

- 1) Помещаем компонент на форму.
- 2) Устанавливаем минимальное и максимальное допустимые значения:

```
WxEntryWord1.MinValue := 10;
WxEntryWord1.MaxValue := 65535;
```

- 3) Записываем на компонент значение и вызываем кнопку со следующим кодом:

```
WxEntryWord1.ValidateEntry();
```

- 4) Если значение внутри диапазона, то ничего не произойдёт. Если значение вне диапазона, то исключительная ситуация «Значение вне диапазона от 10 до 65535». Если значение не целое число, то исключительная ситуация «Неверное значение». Неверные символы (например, минус, буквы или запятая) ввести не удастся.

7.1.3 Компоненты для ввода вещественных чисел

Название	Изображение на палитре компонентов	Изображение на форме при её дизайне	Изображение на форме во время исполнения программы	Назначение
<i>TWxEntryCurrency</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод вещественных чисел типа Currency .
<i>TWxEntryFloat</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод вещественных чисел типа Float .
<i>TWxEntrySingle</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод вещественных чисел типа Single .
<i>TWxEntryExtended</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод вещественных чисел типа Extended .
<i>TWxEntryBCD</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод вещественных чисел типа TBCD .

Пример использования 1:

- 1) Помещаем компонент на форму.
- 2) Заполняем из программы текст компонента:

```
procedure TForm1.Button1Click(Sender: TObject);  
var  
    str0: double;  
begin  
    str0 := 15.25;  
    WxEntryFloat1.SetDataPrim(str0);  
end;
```

В результате на компоненте отобразится 15,25.

Пример использования 2:

- 1) Помещаем компонент на форму.
- 2) Устанавливаем минимальное и максимальное допустимые значения:


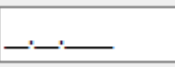
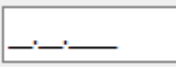

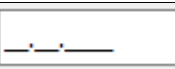
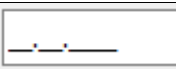

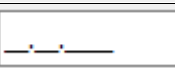
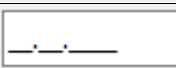

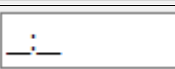
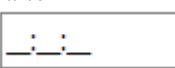

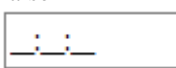
```
WxEntryCurrency1.MinValue := 0.978;  
WxEntryCurrency1.MaxValue := 78978.97;
```

- 3) Записываем на компонент значение и вызываем кнопку со следующим кодом:

```
WxEntryCurrency1.ValidateEntry();
```

- 4) Если значение внутри диапазона, то ничего не произойдёт. Если значение вне диапазона, то исключительная ситуация «Значение вне диапазона от 0,978 до 78978,97». Если значение не денежная величина, то исключительная ситуация «Неверное значение». Неверные символы (например, буквы) ввести не удастся.

7.1.4 Компоненты для ввода дат и времени

Название	Изображение на палитре компонентов	Изображение на форме при её дизайне	Изображение на форме во время исполнения программы	Назначение
<i>TWxEntryDate</i>				Ввод и вывод даты.
<i>TWxEntryDateShortString</i>				Ввод и вывод даты.
<i>TWxEntryDateString</i>				Ввод и вывод даты.
<i>TWxEntryTime</i>		 , если UseLongTimeFormat = false  , если UseLongTimeFormat = true	 , если UseLongTimeFormat = false  , если UseLongTimeFormat = true	Ввод и вывод времени.

Пример использования 1:

- 1) Помещаем компонент *TWxEntryDate*, *TWxEntryDateShortString* или *TWxEntryDateString* на форму.
- 2) Нажимаем «F3».
- 3) В появившейся форме с календарём выбираем требуемую дату.
- 4) В результате на компоненте отобразится выбранное число.

Следует иметь в виду, что «F3» вызывает календарь по умолчанию. Изменить данную клавишу можно путём изменения переменной *WxDictKey* модуля *WxEntry*.

Пример использования 2:

- 1) Помещаем компонент *TWxEntryDateShortString* или *TWxEntryDateString* на форму.
- 2) Устанавливаем свойство *Crazy* на true.
- 3) Вводим на компонент только год.
- 4) Используем метод *ValidateEntry()*, число и месяц заполняются нулями.

При отсутствии пункта 2 возникает исключительная ситуация «Неверное значение». Подобное свойство *Crazy* определено только для компонентов ввода-вывода даты *TWxEntryDateShortString* и *TWxEntryDateString*. Следует иметь в виду, что, если введено только число, месяц и год заполняются автоматически текущим месяцем и годом вне зависимости от свойства *Crazy*.

Пример использования 3:

- 1) Помещаем компонент *TWxEntryDate*, *TWxEntryDateShortString* или *TWxEntryDateString* на форму.
- 2) Нажимаем «F4».
- 3) В результате на компоненте отобразится сегодняшнее число.

Пример использования 4:

- 1) Помещаем компонент *TWxEntryTime* на форму.
- 2) Нажимаем «F4».
- 3) В результате на компоненте отобразится текущее время.

7.1.5 TWxEntryBoolean

Изображение на палитре компонентов: 

Изображение на форме при её дизайне: 

Изображение на форме во время исполнения программы: 

Назначение: ввод и вывод логических переменных.






Пример использования:

- 1) Помещаем на форму компонент TWxEntryBoolean.
- 2) Определяем текстовое представление логических переменных.

```
WxEntryBoolean1.StrTrue := 'OK';
WxEntryBoolean1.StrFalse := 'Cancel';
```

- 3) При выборе из списка логические значения теперь принимают вид «OK» и «Cancel». По умолчанию, эти значения «Да» и «Нет».

7.1.6 Многострочные компоненты для ввода строк

Название	Изображение на палитре компонентов	Изображение на форме при её дизайне	Изображение на форме во время исполнения программы	Назначение
<i>TWxMemoShortString</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод строковых данных типа ShortString .
<i>TWxMemoAnsiString</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод строковых данных типа AnsiString .
<i>TWxMemoWideString</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод строковых данных типа WideString .
<i>TWxMemoUnicodeString</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод строковых данных типа UnicodeString .
<i>TWxMemoString</i>		Совпадает с компонентом Edit (StdCtrls)	Совпадает с компонентом Edit (StdCtrls)	Ввод и вывод строковых данных типа string .

Пример использования:

- 1) Помещаем любой из пяти компонентов на форму под именем *WxMemoString1*.
- 2) Помещаем на форму компонент класса *TEdit Edit1*.
- 3) Нажимаем на кнопку, выполняя код:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  str0: string;
  str: string;
begin
  str0 := 'Database';
  WxMemoString1.SetDataPrim(str0);
```

```
WxMemoString1.GetDataPrim(str);
Edit1.Text := str;
end;
```

4) В результате на компоненте *Edit1* отобразится строка «*Database*».

7.2 Сетки

В библиотеке *WXL* реализована сетка *TWxGrid*. Кроме выполнения стандартной задачи отображения данных заданного набора данных, она также позволяет вызывать всплывающее меню, предоставляющее массу дополнительных возможностей, которые могут понадобиться пользователю при работе с приложением баз данных. Подробная информация о сетке *TWxGrid* может быть получена в главе 5, описывающей интерфейс *WXL*-приложений.

7.2.1 *TWxGrid*

Изображение на палитре компонентов: .

Изображение на форме при её дизайне: совпадает с компонентом *TDbGrid* (*DBGrids*).

Изображение на форме во время исполнения программы: совпадает с компонентом *TDbGrid* (*DBGrids*).

Назначение: Сетка для отображения набора данных, снабжённая всплывающим меню дополнительных возможностей.

Пример использования:

- 1) Помещаем на форму компонент *TWxGrid*.
- 2) Помещаем на форму компонент соединения *ZConnection1* класса *TZConnection*, находящийся на вкладке «*Zeos Access*».
- 3) Помещаем на форму источник данных *DataSource1* класса *TDataSource*, находящийся на вкладке «*Data Access*».
- 4) Помещаем на форму набор данных *ZQuery1* класса *TZQuery*, находящийся на вкладке «*Zeos Access*».
- 5) Помещаем на форму кнопку, и, нажимая на неё, выполняем код:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  ZConnection1.Protocol := 'mssql';
  ZConnection1.HostName := 'COMPOFKIRILL\SQL2016';
  ZConnection1.Database := 'Tourism';
  ZConnection1.LibraryLocation := 'libsybdb-5.dll';
```

```

ZConnection1.ClientCodepage := 'UTF-8';
ZConnection1.User := 'sa';
ZConnection1.Password := '***';

ZQuery1.Connection := ZConnection1;
ZQuery1.SQL.Text := 'select * from DictCity';

DataSource1.DataSet := ZQuery1;
WxGrid1.DataSource := DataSource1;

ZConnection1.Connected := True;
ZQuery1.Open();
end;

```

В данном примере предполагается, что существует база данных *Tourism* под управлением именованного экземпляра СУБД *Microsoft SQL Server* с именем «*COMPOFKIRILL\SQL2016*», в папке с создаваемым исполнимым файлом лежат клиентские библиотеки для соединения с *Microsoft SQL Server*. В результате на компоненте *WxGrid1* отобразится таблица *DictCity* базы данных *Tourism*.

- 6) Нажимаем правую кнопку мыши и выбираем фильтр колонок. Убираем галочки напротив тех столбцов, которые не нужно отображать. Нажимаем «ОК».
- 7) Нажимаем левой кнопкой мыши на центр столбца, порядок которого нужно изменить, перетаскиваем его на требуемую позицию, удерживая левую кнопку мыши (расположится между теми столбцами, между которыми нарисована красная вертикальная черта).
- 8) Нажимаем левой кнопкой мыши на правую границу заголовка столбца, ширину которого необходимо изменить. Удерживая левую кнопку мыши, изменяем размер столбца.

В результате сетка приняла требуемый пользователю вид. Подробная информация о сетке *WxGrid* может быть получена в главе 5, описывающей интерфейс *WXL*-приложений.



7.3 Справочники



Пусть вводимые данные принадлежат заведомо определённом кругу значений. Например, если человек живёт в определённом городе, то улица, на которой он живёт, принадлежит множеству всех улиц этого города. Если требуется ввести страну проживания, то она также может принимать лишь одно из строго

определённых значений. Поэтому имеет смысл обеспечить пользователя, вводящего данные, списком значений, которому эти данные могут принадлежать. С одной стороны, это поможет избежать ошибок ввода, а с другой стороны позволит избежать проблемы указания синонимичных названий. Например, Россия может быть указана и как РФ, Великобритания как Соединённое Королевство или Англия. Для этих целей и были созданы компоненты *WxDictButton* и *WxDictSource*. Компонент *WxDictSource* представляет собой источник словарных данных, а кнопка *WxDictButton* позволяет вызывать присоединённые словарные данные. Кроме того, компоненты ввода-вывода библиотеки *WXL* оснащены удобным механизмом подключения словарных данных. Если такой компонент связан со словарными данными, то можно обеспечить помощь при вводе. То есть, по мере ввода необходимого значения будут предлагаться возможные варианты. Более подробную информацию по работе со справочниками в *WXL*-приложениях можно получить из модуля, описывающего интерфейс *WXL*-приложений.

7.3.1 *TWxLinkButton*

Изображение на палитре компонентов: .

Изображение на форме при её дизайне:  или кнопка с выбранной картинкой (например, ).

Изображение на форме во время исполнения программы:  или кнопка с выбранной картинкой (например, .

Назначение: Базовый компонент для кнопки вызова словаря.

Пример использования:

- 1) Помещаем компонент *TWxLinkButton* на форму.
- 2) Помещаем на форму компонент класса *TWxEntryString WxEntryString1*.
- 3) Помещаем на форму кнопку, и, нажимая на неё, выполняем код:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Picture: TPicture;
begin
  Picture := TPicture.Create();
  try
    Picture.LoadFromFile('wxb_key.png');
    WxLinkButton1.Glyph.Assign(Picture.Bitmap);
  end;
```

```


WxLinkButton1.LinkAttach := abRight;
WxLinkButton1.LinkSeparation := 1;
WxLinkButton1.LinkControl := WxEntryString1;
finally
  Picture.Free();
end;
end;


```

- 4) В результате справа от компонента *WxEntryString1* расположится кнопка *WxLinkButton1*, имеющая вид кнопки с нарисованной картинкой «*wxb_key.png*».

7.3.2 *TWxDictButton*

Изображение на палитре компонентов: .

Изображение на форме при её дизайне:  или кнопка с выбранной картинкой.

Изображение на форме во время исполнения программы:  или кнопка с выбранной картинкой.

Назначение: Вызов присоединённых словарей.

Пример использования:

- 1) Помещаем на форму компонент *WxDictButton1* класса *TWxDictButton*.
- 2) Помещаем на форму компонент класса *TWxEntryString* *WxEntryString1*.
- 3) Помещаем на форму невидимые компоненты *ZConnection1* класса *TZConnection* и *ZQuery1* класса *TZQuery* с вкладки «*Zeos Access*». Данные компоненты необходимы, так данные словаря берутся из базы данных выбранной СУБД (в рассматриваемом примере это *Microsoft SQL Server*).
- 4) Помещаем на форму компонент *WxDictSource1* класса *TWxDictSource*.
- 5) Помещаем на форму кнопку, и, нажимая на неё, выполняем код:

```

procedure TForm1.Button1Click(Sender: TObject);
begin
  {Разрешить подсказки при вводе.}
  WxEntry.WxIncrementalDictSearch := true;

  {Очистка компонента перед началом работы.}
  WxEntryString1.Text := '';

  ZConnection1.Protocol := 'mssql';
  ZConnection1.HostName := 'COMPOFKIRILL\SQL2016';
  ZConnection1.Database := 'Tourism';
  ZConnection1.LibraryLocation := 'libsybdb-5.dll';
  ZConnection1.ClientCodepage := 'UTF-8';
  ZConnection1.User := 'sa';

```

```

ZConnection1.Password := '***';

ZQuery1.Connection := ZConnection1;
ZQuery1.SQL.Text := 'select * from DictCity';

{Набор данных для источника данных словаря = записи таблицы, удовлетворяющие
запросу.}
WxDictSource1.DataSet := ZQuery1;
{В качестве источника данных используется источник словарных данных библиотеки
WXL.}
WxDictButton1.DictSource := WxDictSource1;
{При отсутствии данной строки подсказки при вводе будут недоступны до первого
ручного нажатия WxDictButton1.}
WxDictSource1.KeyFieldName := 'CityName';
{При вводе данных на WxEntryString1 будут предлагаться варианты окончания.
Каждый выбор добавляет новое значение на компонент WxEntryString1 через
запятые.}
WxDictButton1.DictOptions:= [dictLookup, dictConcat];
{На форме выбора значений из словаря имеются кнопки "OK" и "Отмена".}
WxDictStyle := dsFormOkCancel;
{Кнопка располагается рядом с компонентом.}
WxDictButton1.LinkControl := WxEntryString1;

ZConnection1.Connected := True;
ZQuery1.Open();
end;

```

Назначения некоторых строк выделены примечаниями перед строчками кода.

- б) В результате при вводе данных на компонент *WxEntryString1* будут всплывать подсказки. Кроме того, нажав стрелку вниз или «F3» можно вызвать форму словаря и выбрать требуемое значение, которое будет автоматически занесено на компонент. Вместо «F3» можно использовать любую другую клавишу, меняя значение переменной *WxDictKey* модуля *WxEntry*. Стрелка вниз будет срабатывать только при условии *WxUpDownAsTab false*, а вместо самой стрелки вниз можно использовать любую другую клавишу, меняя значение переменной *WxDictKey2*.

7.3.3 *TWxDictSource*

Изображение на палитре компонентов:  .

Изображение на форме при её дизайне:  .

Изображение на форме во время исполнения программы: Невизуальный компонент.

Назначение: Источник данных для присоединённых словарей.

Пример использования:

- 1) Помещаем на форму компонент *WxDictSource1* класса *TWxDictSource*.
- 2) Помещаем на форму компонент *WxEntryString1* класса *TWxEntryString*.
- 3) Помещаем на форму невидимые компоненты *ZConnection1* класса *TZConnection* и *ZQuery1* класса *TZQuery* с вкладки «Zeos Access». Данные компоненты необходимы, так данные словаря берутся из базы данных выбранной СУБД (в рассматриваемом примере это *Microsoft SQL Server*).
- 4) Помещаем на форму компонент *WxDictButton1* класса *TWxDictButton*.
- 5) Помещаем на форму кнопку, и, нажимая на неё, выполняем код:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  {Разрешить подсказки при вводе.}
  WxEntry.WxIncrementalDictSearch := true;

  {Очистка компонента перед началом работы.}
  WxEntryString1.Text := '';

  ZConnection1.Protocol := 'mssql';
  ZConnection1.HostName := 'COMPOFKIRILL\SQL2016';
  ZConnection1.Database := 'Tourism';
  ZConnection1.LibraryLocation := 'libsybdb-5.dll';
  ZConnection1.ClientCodepage := 'UTF-8';
  ZConnection1.User := 'sa';
  ZConnection1.Password := '***';

  ZQuery1.Connection := ZConnection1;
  ZQuery1.SQL.Text := 'select * from DictCity';

  {Набор данных для источника данных словаря = записи таблицы, удовлетворяющие
запросу.}
  WxDictSource1.DataSet := ZQuery1;
  {В качестве источника данных используется источник словарных данных библиотеки
WXL.}
  WxDictButton1.DictSource := WxDictSource1;
  {При отсутствии данной строки подсказки при вводе будут недоступны до первого
ручного нажатия WxDictButton1.}
  WxDictSource1.KeyFieldName := 'CityName';
  {При вводе данных на WxEntryString1 будут предлагаться варианты окончания.
Каждый выбор добавляет новое значение на компонент WxEntryString1 через запятые.}
  WxDictButton1.DictOptions:= [dictLookup, dictConcat];
  {На форме выбора значений из словаря имеются кнопки "OK" и "Отмена".}
  WxDictStyle := dsFormOkCancel;
  {Кнопка располагается рядом с компонентом.}
  WxDictButton1.LinkControl := WxEntryString1;

  ZConnection1.Connected := True;
  ZQuery1.Open();
end;
```


Назначения некоторых строк выделены примечаниями перед строчками кода.

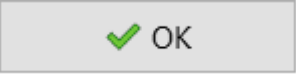
- б) В результате при вводе данных на компонент *WxEntryString1* будут всплывать подсказки. Кроме того, нажав стрелку вниз или «F3» можно вызвать форму словаря и выбрать требуемое значение, которое будет автоматически занесено на компонент. Вместо «F3» можно использовать любую другую клавишу, меняя значение переменной *WxDictKey* модуля *WxEntry*. Стрелка вниз будет срабатывать только при условии *WxUpDownAsTab false*, а вместо самой стрелки вниз можно использовать любую другую клавишу, меняя значение переменной *WxDictKey2*.

7.4 Дополнительные кнопки

7.4.1 *TWxBitBtn*

Изображение на палитре компонентов: .

Изображение на форме при её дизайне: обычная кнопка с надписью или кнопка с выбранной картинкой и надписью (например, ).

Изображение на форме во время исполнения программы: обычная кнопка с надписью или кнопка с выбранной картинкой и надписью (например, ).



Назначение: кнопка с картинкой и надписью, картинка может выбираться из списка стандартных *WXL*-картинок.



Пример использования:

- 1) Помещаем компонент *WxBitBtn1* класса *TWxBitBtn* на форму.
- 2) Выбираем картинку *btn_ok* из списка *GlyphResName*.
- 3) В результате на кнопке *WxBitBtn1* появилась картинка в виде зелёной галочки.

7.4.2 *TWxSpeedButton*

Изображение на палитре компонентов: .

Изображение на форме при её дизайне:  или кнопка с выбранной картинкой (например, ).

Изображение на форме во время исполнения программы:  или кнопка с выбранной картинкой (например, ).

Назначение: кнопка с картинкой, картинка может выбираться из списка стандартных *WXL*-картинок.

Пример использования:

- 1) Помещаем компонент *WxSpeedButton1* класса *TWxSpeedButton* на форму.
- 2) Выбираем картинку *wxb_key* из списка *GlyphResName*.
- 3) В результате на кнопке *WxSpeedButton1* появилась картинка в виде ключа.

7.5 Особенности использования компонента *TWxTransfer*

Изображение на палитре компонентов: 

Изображение на форме при её дизайне: 

Изображение на форме во время исполнения программы: Невизуальный компонент.

Назначение: Связь компонентов ввода-вывода данных.

Пример использования:

- 1) Помещаем компонент *WxTransfer1* класса *TWxTransfer* на форму.
- 2) Помещаем на форму компоненты *WxEntryDate1*, *WxEntryTime1* и *WxMemoString1*.
- 3) Помещаем на форму кнопку, и, нажимая на неё, выполняем код:

```
procedure TForm1.Button1Click(Sender: TObject);
type
  TTestData = record
    FDateTime1: TDate;
    FDateTime2: TTime;
    FString: String;
  end;

var
  InputRecord: TTestData;
begin
  WxTransfer1.Add(WxEntryDate1);
  WxTransfer1.Add(WxEntryTime1);
  WxTransfer1.Add(WxMemoString1);
```

```
InputRecord.FDateTime1 := StrToDate('10.10.2003');
InputRecord.FDateTime2 := StrToTime('12:12:35');
InputRecord.FString := 'Insert';
WxTransfer1.SetData(InputRecord);
end;
```

В результате на компоненте *WxEntryDate1* отобразится «10.10.2003», на компоненте *WxEntryTime1* отобразится «12:12:35» или «12:12» в зависимости от свойства *UseLongTimeFormat*, на компоненте *WxMemoString* отобразится слово «*Insert*».

7.6 Манипулирование данными в *wxl*-приложениях

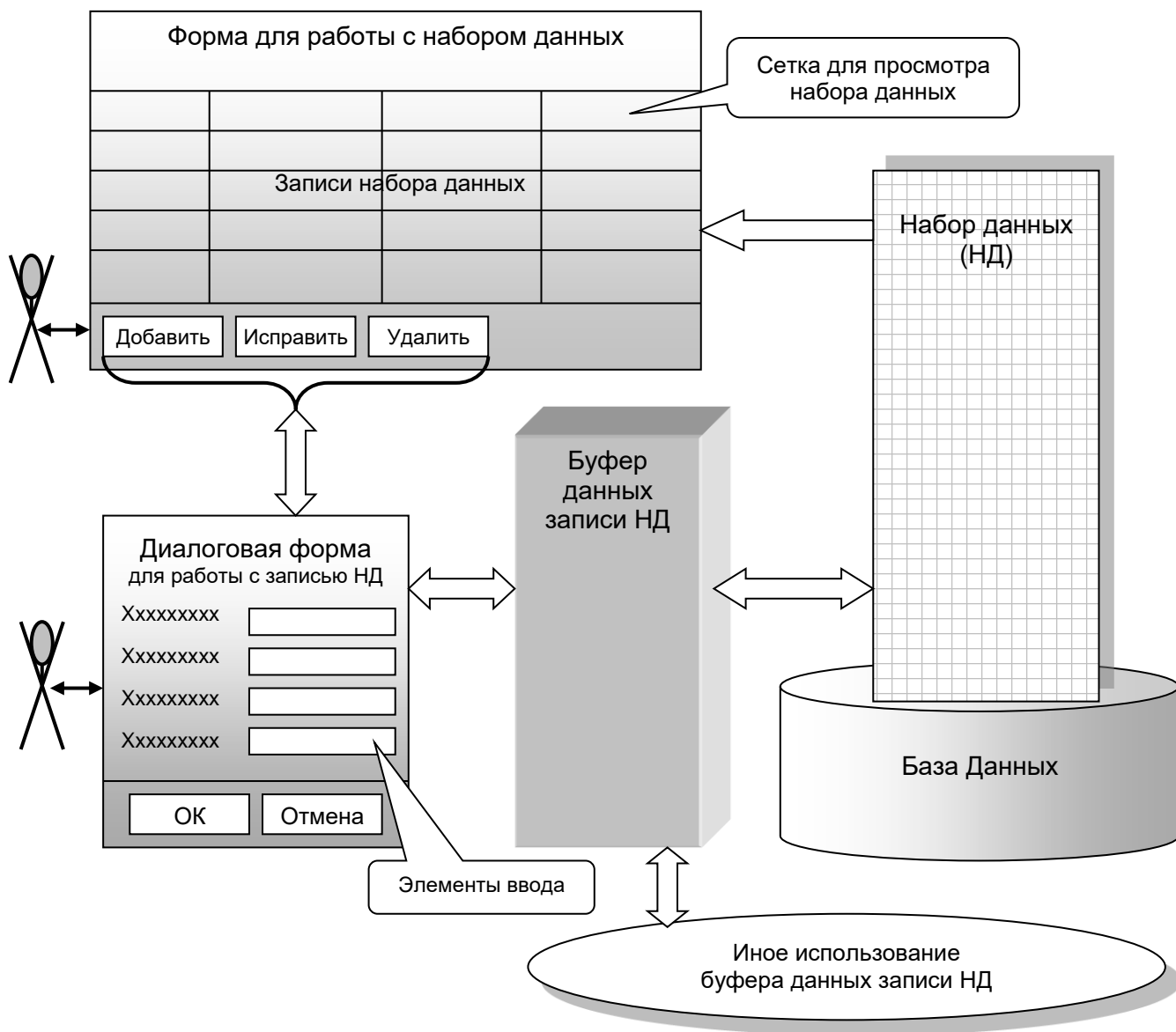
Основная задача *WXL*-приложений – это работа с базами данных. Практически во всех подобных приложениях возникает набор типовых задач:

- 1) Выборка информации из базы данных по заданным критериям.
- 2) Визуализация наборов данных и навигация по ним.
- 3) Представление записей наборов данных в виде диалоговых форм.
- 4) Ввод, модификация и удаление записей наборов данных.

Понятно, что решение всех этих задач не сводится к одному только интерфейсу конечного пользователя. Следовательно, необходимо создание продуманной технологии и архитектуры реализации перечисленных операций. Заблаговременное планирование архитектуры создаваемого приложения баз данных с одной стороны повышает его надёжность, а с другой позволяет значительно увеличить скорость работы приложения благодаря созданию так называемых генераторов исходного кода и ресурсов приложения.

Отсюда мы приходим к необходимости использовать *WXL*-технологию, которая основана на библиотеке *WXL*, и генератор *wxlgen*, позволяющий автоматически создавать каркас типового *WXL*-приложения. *WXL*-технология использовалась в целой серии проектов, начиная с 1998 года, и доказала свою эффективность в разработке высоконадёжных современных информационных систем. Кроме того, следует указать, что совместное использование библиотеки *WXL* с компонентами *RSX*-синхронизации, используемыми для обмена информацией в распределённых базах данных, предоставляет разработчикам дополнительные преимущества.

Рассмотрим, как *WXL*-технология решает типовые задачи, возникающие при создании приложений баз данных. Генератор *wxlgen* создаёт специальные записи, которые полностью соответствуют кортежам таблиц, с которыми происходит работа. Для работы с каждой таблицей создаётся модуль и форма, содержащие функциональность по добавлению, исправлению и удалению записей в таблице. Для осуществления добавления и исправления записей в таблице вызываются созданные генератором диалоги, в которых компоненты ввода-вывода связаны в единую структуру при помощи компонента *TWxTransfer*. Сама сетка *TWxGrid* позволяет использовать множество дополнительных возможностей, которые могут потребоваться пользователю при работе с набором данных. Вызов этих возможностей осуществляется при помощи вызова присоединённого к компоненту всплывающего меню. Ещё одной особенностью *WXL*-технологии является использование типизированных наборов данных, предназначенных для работы с конкретными таблицами. Следует иметь в виду, что для самих диалогов ввода и редактирования записей таблицы создаются типизированные компоненты ввода-вывода, что позволяет легко задавать ограничения, наложенные на поля таблицы, а также присоединять словари, если значения выбираются из определённого набора. Вот как выглядит схема работы с данными при использовании *WXL*-технологии:



Типовые задачи приложений баз данных

7.7 GUI-модули для манипулирования данными

- 1) Модуль *WxButtons* содержит реализацию WXL-компонентов для кнопок: *TWxBitBtn*, *TWxSpeedButton*, *TWxLinkButton*.
- 2) Модуль *WxDataSetExportFrm* содержит реализацию формы для экспорта набора данных в файл.
- 3) Модуль *WxDataSetSort* содержит классы и функции, ориентированные на сортировку извлечённых из базы данных таблиц по заданным пользователем столбцам. Используется модулем *WxDbGrid*.
- 4) Модуль *WxDataSetView* содержит класс и функции, ориентированные на просмотр таблиц баз данных.

- 5) Модуль *WxEntry* содержит классы и функции, предназначенные для реализации компонентов ввода-вывода данных.
- 6) Модуль *WxExportParamsPanelCSV* содержит реализацию панели для выбора параметров экспорта в *CSV*-формате. Является вспомогательным для модуля *WxDataSetExportFrm*.
- 7) Модуль *WxExportParamsPanelSQL* содержит реализацию панели для выбора параметров экспорта в *SQL*-формате. Является вспомогательным для модуля *WxDataSetExportFrm*.
- 8) Модуль *WxTblDict* содержит классы и функции, ориентированные на работу со словарём.
- 9) Модуль *WxDbGrid* содержит реализацию визуального компонента *WxDbGrid*.
- 10) Модуль *WxDbGridDlg* является вспомогательным для модуля *WxDbGrid*. В нём содержится реализация опций, вызываемых по всплывающему *PopupMenu WxGrid* по умолчанию.

7.8 Модули без GUI для манипулирования данными

- 1) Модуль *WxDataSet* содержит функциональность по созданию различных наборов данных в памяти на основе *TDataSet*.
- 2) Модуль *WxDataSet2BDS* содержит функциональность по конвертации набора данных *TDataSet* в специальный бинарный *BDS*-формат.
- 3) Модуль *WxDataSet2KBM* содержит функциональность по конвертации набора данных *TDataSet* в *KBM*-формат.
- 4) Модуль *WxDataSet2VTD* содержит функциональность по конвертации набора данных *TDataSet* в *VTD*-формат компонента *VirtualTable* в составе *UniDAC*.
- 5) Модуль *WxDataSetExport* является вспомогательным для модуля *WxDbGrid*. Он предназначен для экспорта набора данных в файл выбранного формата. Поддерживаемые форматы: *TXT, XLS, HTM, XML, ADT, DB, DBF, DAT*.
- 6) Модуль *WxDataSetInfo* предназначен для получения информации о наборе данных, используется как вспомогательный для модуля *WxDataSetView*.
- 7) Модуль *WxDataSetStorageBIN* предназначен для сохранения и загрузки набора данных в бинарном формате.

- 8) Модуль *WxDataSetStorageJSON* предназначен для сохранения и загрузки набора данных в формате *JSON*.
- 9) Модуль *WxDataSetStorageText* предназначен для сохранения и загрузки набора данных в текстовом формате.
- 10) Модуль *WxDataSetStorageXML* предназначен для сохранения и загрузки набора данных в формате *XML*.
- 11) Модуль *WxExportParams* содержит классы для параметров экспорта. Является вспомогательным для модулей *WxDataSetExport* и *WxDataSetExportFrm*.
- 12) Модуль *WxDataSetScan* является вспомогательным для модуля *WxDbGridDlg*. В нём содержится реализация средств для сканирования таблиц, которые затем используются для фильтрации и поиска записей таблицы, отвечающих набору условий по столбцам.
Модуль *WxDbCommon* содержит вспомогательную функциональность для модулей *WxDbGrid* и *WxDbGridDlg*.

8. Генерация отчётности

8.1 Общая информация

Одна из самых распространённых операций, которую проделывают приложения по работе с базами данных, – это распечатка отчётов. Отчёты (Reports) представляют собой печатные документы, содержащие информацию из базы данных. Их основное назначение – предоставить удобное средство анализа данных. Для создания отчётов чаще всего используются специализированные программы или наборы компонентов – генераторы отчётов. Чаще всего они позволяют включать в отчёт вычисляемые значения (например, сумма каких-то величин) и простейшую бизнес-графику. Когда требуются более сложные средства анализа данных, то необходимо использовать OLAP (online analytical processing). Вместе с *Lazarus* поставляется генератор отчётов *LazReport*. В библиотеке *WXL* для генерации отчётов можно использовать либо *LazReport*, либо *FastReport* фирмы *Fast Reports Inc.* Среди других генераторов отчёта можно указать также *Crystal Reports* фирмы *SAP*. Существует три основных типа отчётов:

- 1) Табулированные отчёты (*tabular reports*) – каждой строке отчёта соответствует запись набора данных.
- 2) Отчёты-этикетки – каждой записи набора данных соответствуют прямоугольные области одинаковой ширины и высоты.
- 3) Отчёты типа «главный-подчинённый» – содержатся данные из связанных наборов данных.

Для генераторов отчётов *FastReport* и *LazReport* характерно деление отчёта на полосы (bands). Каждая полоса соответствует своей части отчёта. Типовая структура отчёта включает следующие полосы:

- 1) *Page Header* – заголовок страницы. Данная полоса располагается в начале каждой страницы отчёта и содержит служебную информацию об отчёте.
- 2) *Title* – заголовок отчёта. Данная полоса располагается на первой странице отчёта и непосредственно следует за заголовком страницы. Как и следует из названия, она содержит заголовок отчёта.

- 3) *Column Header* – заголовок столбца. Данная полоса содержится в любом отчёте многократно. Через её функциональность обеспечивается описание, что за данные отображаются.
 - 4) *Detail* – полоса данных. Данная полоса содержит данные, которые необходимо представить в отчёте.
 - 5) Краткое описание – итоговая полоса. Данная полоса располагается после полосы данных и содержит вычисляемые значения, относящиеся к данным (например, сумма каких-то величин).
 - 6) *Page Footer* – полоса окончания страницы. Данная полоса располагается в конце каждой страницы отчёта и содержит служебную информацию об отчёте.
- Приведём пример сгенерированного средствами *FastReport* отчёта:

Дубликаты в реестре туроператоров

22.04.2022 14:20:30

[COMPOFKRILL\SQL2016] [SQL.Consul\DKS]

Статус компаний Действующие
Не действующие

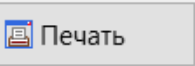
ID	Номер референса	Действует до	Руководитель	Адрес	Телефон	Примечания	Дата	Актуально (0/1)
ВЛАДИМИР / ООО ВЛАДИМИРСКОЕ ГОРОДСКОЕ БЮРО ПУТЕШЕСТВИЙ								
8233	016885	31.05.2018	-	600022, г. Владимир, пр-т Ленина, 48	-	-	12.12.2017 17:28:34	1
8277	020387	31.07.2019	-	600022, г. Владимир, проспект Ленина, д.48, комн. 29, 39	-	-	26.09.2018 18:02:48	1
ЕВПАТОРИЯ / ООО ТУРФИРМА ЛАГУНА-КРЫМ								
5082	016074	10.11.2018	-	297412, Евпатория, ул.Шевченко, д.37, оф.4	-	-	24.12.2019 15:57:10	1
6649	017999	10.11.2018	-	297406, Республика Крым, г. Евпатория, ул.Фрунзе, д.83, кв.17	-	-	13.12.2017 15:29:59	1
ЕКАТЕРИНБУРГ / ООО ТУР-БАН И К								
6106	017262	31.05.2017	-	620137, г. Екатеринбург, ул. Галарина, д. 27, кв. 28	-	-	08.06.2016 10:18:09	1
6847	018071	31.10.2017	-	620014, Екатеринбург, пер. Театральный, 2	-	-	09.11.2016 10:38:52	1
ЕКАТЕРИНБУРГ / ООО ЮНИТУР-2007								
8368	017614	28.10.2017	-	620075, Екатеринбург, ул.Пушкина, 5, лит.Б	-	-	11.08.2018 11:59:03	1
8581	020781	31.01.2020	-	620075, г. Екатеринбург, ул. Пушкина, д.5	-	-	07.02.2019 10:20:13	1
ИРКУТСК / ООО АММИКАН								
7705	019864	07.12.2018	-	664009, г.Иркутск, ул. Идринцевод, 15, кв. 2	-	-	24.01.2018 16:35:53	1
8575	020800	27.01.2020	-	664009, г.Иркутск, ул. Ядринцева, д.15, кв. 2	-	-	14.02.2019 09:37:20	1
ИРКУТСК / ООО БАЙКАЛЬСКИЙ ЦЕНТР ТУРИЗМА								
4954	013516	25.07.2015	-	664047, Иркутск, Советская, д.27	-	-	09.10.2014 16:35:53	1
8166	020190	24.05.2019	-	664047, г.Иркутск, ул. Ал. Невского, д.58, оф. 22	-	-	18.07.2018 17:58:03	1

Стр. 1 из 11

ID	Номер референса	Действует до	Руководитель	Адрес	Телефон	Примечания	Дата	Актуально (0/1)
ИРКУТСК / ООО БМТК								
6742	018221	14.11.2017	-	664019, Иркутск, пер. Сирова, 10а	-	-	06.12.2016 16:24:31	1

В качестве примера использования *FastReport* приведём его использование в сгенерированном при помощи генератора *wxlgen* приложении.

В типовом *WXL*-приложении в форме, предназначенной для работы с заданной таблицей, существует два способа построения отчёта:

- 1) Использование кнопки  Печать.
 - 2) Нажатие правой кнопкой мыши на сетке и выбор во всплывающем меню пункта «Печать».
- Вот как выглядит эта форма:

ID	Номер референса	Действует до	Название компании	Населённый пункт	Сокр.Назв.Компании
1324	000001	08.02.2018	ООО АКАДЕМСЕРВИС	МОСКВА	ООО АКАДЕМСЕРВИС
5561	000008	27.09.2018	АО СКО РОСЮГУРТОРТ	СОЧИ	АО СКО РОСЮГУРТОРТ
1355	000063	31.05.2017	ООО МИНТУР	МУРМАНСК	ООО МИНТУР
1648	000067	31.05.2018	ООО ДРУЖБА	ОЛОНЕЦ	ООО ДРУЖБА
1329	000068	31.05.2018	ООО ХОТЭЛ МЕНЕДЖМЕНТ КОМПАНИ	МОСКВА	ООО ХОТЭЛ МЕНЕДЖМЕНТ КОМ
1164	000069	31.07.2018	ООО МАКСИМА ХОТЕЛС	МОСКВА	ООО МАКСИМА ХОТЕЛС
126	000072	31.05.2018	ООО НОРДВЭЙ ИК	МОСКВА	ООО НОРДВЭЙ ИК
1363	000074	31.05.2018	ООО ПРОФСЕРВИС	ЕКАТЕРИНБУРГ	ООО ПРОФСЕРВИС
1649	000075	31.05.2018	ООО ТФ КАПРИЗ	САНКТ-ПЕТЕРБУРГ	ООО ТФ КАПРИЗ
263	000076	15.10.2019	АО ГОСТИНИЦА СОВЕТСКАЯ	САНКТ-ПЕТЕРБУРГ	АО ГОСТИНИЦА СОВЕТСКАЯ
301	000077	16.09.2018	ПАО ЦЕНТР МЕЖДУНАРОДНОЙ ТОРГОВЛИ	МОСКВА	ПАО ЦМТ
1216	000080	31.05.2018	ООО ТУРИСТИЧЕСКАЯ ФИРМА КРАСНОВ	ПЕРМЬ	ООО ТФ КРАСНОВ
1038	000084	18.06.2018	ООО АЛЬВИСТ	МОСКВА	ООО АЛЬВИСТ
1220	000085	22.07.2018	ООО ГЛОРИАНН	МОСКВА	ООО ГЛОРИАНН
1009	000086	15.10.2018	ООО ПЕРЕСВЕТ ТУР	МОСКВА	ООО ПЕРЕСВЕТ ТУР
1053	000087	31.05.2018	ООО ТОРГОВЫЙ ДОМ МЕНАХЕМ	САНКТ-ПЕТЕРБУРГ	ООО ТОРГОВЫЙ ДОМ МЕНАХЕМ
230	000088	31.05.2018	ООО АЛИРА-ТУР	ВЛАДИВОСТОК	ООО АЛИРА-ТУР
1361	000114	31.05.2018	ООО СЛАВЯНСКАЯ	МОСКВА	ООО СЛАВЯНСКАЯ
1242	000115	01.08.2018	ЗАО НЕВАЛЬ	САНКТ-ПЕТЕРБУРГ	ЗАО НЕВАЛЬ
1112	000118	01.06.2018	ЗАО СЕВЕРНЫЕ РЕКИ	СЕВЕРОМОРСК	ЗАО СЕВЕРНЫЕ РЕКИ
299	000119	01.06.2018	ООО ХАРЛОВКА	МУРМАНСК	ООО ХАРЛОВКА
513	000120	31.05.2018	ЗАО ГК АСТОРИЯ	САНКТ-ПЕТЕРБУРГ	ЗАО ГК АСТОРИЯ
48	000121	31.05.2018	ЗАО СТАР ТРЭВЕЛ ЛТД	САНКТ-ПЕТЕРБУРГ	ЗАО СТАР ТРЭВЕЛ ЛТД
1653	000123	08.06.2018	ООО РАДИАН-ТУР	ИРКУТСК	ООО РАДИАН-ТУР

Компоненты генераторов отчётов *LazReport* и *FastReport* можно разделить на три группы:

- 1) Компоненты размещения – представляют собой компоненты-контейнеры для компонентов вывода на печать.
- 2) Компоненты вывода на печать – служат для вывода на печать информации отчёта.
- 3) Компоненты экспорта данных – служат для экспорта данных из предопределённого формата отчёта в другие форматы.

Базовым форматом отчётов *LazReport* и *FastReport* является формат «*frp*». Генератор отчётов *FastReport* позволяет экспортировать отчёт в форматы *BMP*, *JPG*, *TIF*, *PNG*, *TXT*, *CSV*, *RTF*, *ODT*, *ODS*, *HTML*, *DOCX*, *XLSX*, *PPTX*, *PDF*, *XML*, *ZPL*, *PS*, *PPML*, *DBF*. Генератор отчётов *LazReport* позволяет экспортировать отчёт в форматы *TXT*, *CSV*, *HTML*, *PDF*.

Для генерации отчётов с использованием библиотеки *WXL* предназначен модуль *WxTblRpt*. В данном модуле содержатся функции создания отчёта для набора данных и для сетки. Если необходимо создать отчёт по набору данных, то можно использовать функцию *RptDataSetProcess*. Если необходимо создать отчёт по сетке с данными, то необходимо использовать функцию *RptGridProcess*. Если использована директива условной компиляции *USEFASTREPORT*, то будет

использован генератор отчётов *FastReport*. В этом случае будет вызываться вспомогательный модуль *WxTblRptFR6*. Если использована директива условной компиляции *USELAZREPORT*, то будет использован генератор отчётов *LazReport*. В этом случае будет вызываться вспомогательный модуль *WxTblRptL*.

8.2 Модули для работы с отчётностью

Модули для работы с отчётностью размещены в подпапке *rpt* исходников библиотеки *WXL*. Далее приведён их полный список:

- 1) Модуль *DlgWxTblRpt* содержит реализацию формы для ввода параметров отчёта.
- 2) Модуль *WxTblRpt* ориентирован на создание отчётов *LazReport* или *FastReport* по набору данных.
- 3) Модуль *WxTblRptExp* – модуль данных для экспорта отчётов.
- 4) Модуль *WxTblRptFR6* ориентирован на создание отчётов по набору данных с использованием *FastReport*.
- 5) Модуль *WxTblRptFR6Exp* ориентирован на экспорт отчётов *FastReport*.
- 6) Модуль *WxTblRptL* ориентирован на создание отчётов по набору данных с использованием *LazReport*.
- 7) Модуль *WxTblRptLExp* ориентирован на экспорт отчётов *LazReport*.

9. Модули для работы с ОС

- 1) *GUI*-модули для работы с операционной системой и файловой системой (размещены в подпапке *gui* исходников библиотеки *WXL*).
 - 1.1) Модуль *WxFileDialog* содержит диалоги для выбора файлов из файловой системы.
 - 1.2) Модуль *WxFileManagerDlg* содержит реализацию диалога для выполнения стандартных операций через файловый менеджер: копирование, перемещение, создание каталога, переименование и удаление.
 - 1.3) Модуль *WxFileManagerFra* содержит реализацию панели двухпанельного файлового менеджера.
 - 1.4) Модуль *WxFileManagerFrm* содержит реализацию основной формы двухпанельного файлового менеджера.
 - 1.5) Модуль *WxFileManagerInterface* содержит реализацию программного интерфейса для двухпанельного файлового менеджера.
 - 1.6) Модуль *WxFileManagerListView* содержит реализацию просмотрщика набора данных для хранения списка файлов на основе *TCustomDbGrid*.
 - 1.7) Модуль *WxTXTView* содержит реализацию формы просмотра текстовых файлов.
- 2) Модули без *GUI* для работы с операционной системой и файловой системой.
 - 2.1) Модуль *WxClasses* содержит реализацию классов *TADFileStream* и *TStreamWriter*.
 - 2.2) Модуль *WxFileAttr* содержит функциональность для работы с атрибутами файлов.
 - 2.3) Модуль *WxFileManagerList* содержит реализацию набора данных на основе *TWxMemDataSet* для хранения списка файлов.
 - 2.4) Модуль *WxFiles* содержит процедуры и функции для выполнения стандартных действий над файлами и папками в файловой системе (копирование, переименование, перемещение, ...).
 - 2.5) Модуль *WxFileTime* содержит функции для просмотра и установки атрибутов даты создания, изменения и последнего доступа к файлам.
 - 2.6) Модуль *WxFileUtils* содержит функциональность по сохранению и загрузке строк из потоков (*TStream*).

- 2.7) Модуль *WxFileVer* содержит функциональность по работе с версией продукта *Lazarus* (*Major, Minor, Release, Build*): получение версии продукта, сравнение версии продукта у двух заданных файлов.
- 2.8) Модуль *WxINIFiles* содержит функциональность по работе с *INI*-файлами.
- 2.9) Модуль *WxSysUtils* содержит несколько функций: получение имени компьютера, модуля и пользователя; получение доступной и используемой памяти.

10. Взаимодействие с СУБД

10.1 Поддерживаемые СУБД и модуль *wxtypes*

Модуль *WxTypes* содержит набор констант и типов для выбора СУБД, с которыми будет осуществляться работа. В частности, содержатся функции для преобразований различных представлений типов *SQL*-серверов: строкового (первый столбец таблицы), целочисленного (третий столбец таблицы), перечислимого типа *TWxSQLServerType*.

СУБД	Константа	Значение	<i>TWxSQLServerType</i>	Подсветка <i>WxSynEdit</i> : <i>TWxSynEditDialect</i>
MSSQL	<i>intMSSQL</i>	0	<i>stMSSQL</i>	<i>seMSSQL</i>
Oracle	<i>intORACLE</i>	10	<i>stORACLE</i>	<i>seORACLE</i>
Sybase ASE	<i>intASE</i>	20	<i>stASE</i>	<i>seASE</i>
Sybase Anywhere	<i>intASA</i>	30	<i>stASA</i>	<i>seASA</i>
Sybase IQ	<i>intIQ</i>	90	<i>stIQ</i>	<i>seIQ</i>
Interbase	<i>intINTERBASE</i>	40	<i>stINTERBASE</i>	<i>seINTERBASE</i>
Firebird	<i>intFIREBIRD</i>	120	<i>stFIREBIRD</i>	<i>seFIREBIRD</i>
MySQL	<i>intMYSQL</i>	50	<i>stMYSQL</i>	<i>seMYSQL</i>
Lintier	<i>intLINTER</i>	60	<i>stLINTER</i>	<i>seLINTER</i>
DB2	<i>intDB2</i>	70	<i>stDB2</i>	<i>seDB2</i>
ElevateDB	<i>intEDB</i>	80	<i>stEDB</i>	<i>seEDB</i>
PostgreSQL	<i>intPOSTGRESQL</i>	100	<i>stPOSTGRESQL</i>	<i>sePOSTGRESQL</i>
Informix	<i>intINFORMIX</i>	110	<i>stINFORMIX</i>	<i>seINFORMIX</i>
			<i>stUnknown</i>	<i>seNone</i> , <i>seSQL</i> , <i>sePAS</i> , <i>seLFM</i> , <i>seXML</i>

Перечислим важные функции модуля *WxTypes*:

- 1) *SQLServerTypeToStr* => преобразование из *TWxSQLServerType* в строку.
- 2) *StrToSQLServerType* => преобразование из строки в *TWxSQLServerType*.
- 3) *SQLServerTypeToInt* => преобразование из *TWxSQLServerType* в число.
- 4) *IntToSQLServerType* => преобразование из числа в *TWxSQLServerType*.
- 5) *SQLServerTypeToSynEditDialect* => преобразование из *TWxSQLServerType* в *TWxSynEditDialect*.

10.2 GUI-модули для взаимодействия с различными СУБД

- 1) Модуль *FrmDbIndexes* служит для отображения индексов таблиц, принадлежащих БД. Является вспомогательным для модуля *FrmDbTables* через модуль *WxDBHelp*.

- 2) Модуль *FrmDbOptions* предоставляет доступ к опциям БД, а также позволяет изменить их. Данный модуль является вспомогательным для модуля *FrmSQLEx*, хотя может быть использован самостоятельно.
- 3) Модуль *FrmDbProcedures* служит для отображения хранимых на сервере процедур, функций и представлений БД. Является вспомогательным для модуля *FrmSQLEx* через модуль *WxDBHelp*.
- 4) Модуль *FrmDbTables* служит для отображения названий таблиц, принадлежащих БД. Кроме того, позволяет работать с этими таблицами, получая информацию об их содержимом, структуре, индексах, триггерах.
- 5) Модуль *FrmDbTriggers* служит для отображения триггеров таблиц, принадлежащих БД. Является вспомогательным для модуля *FrmDbTables* через модуль *WxDBHelp*.
- 6) Модуль *FrmSQLEx* при помощи функции *ExecuteSQLForm* вызывает анализатор SQL-запросов, подобный *Management Studio* для *Microsoft SQL Server*.
- 7) Модуль *WxDBHelpGUI* – реализация интерфейса для получения информации об объектах БД. Является вспомогательным для модулей: *FrmDBIndexes*, *FrmDBOptions*, *FrmDBTables*, *FrmDBTriggers*, *FrmDBProcedures*.
- 8) Модуль *WxDBSQLBackUpFrm* – реализация формы для резервного копирования данных.
- 9) Модуль *WxDBSQLPrmFrm* содержит реализацию формы *TFormSQLParams* для отображения SQL-параметров заданного SQL-запроса.
- 10) Модуль *WxObjectInspector* является вспомогательным для модуля *FrmSQLEx*. Позволяет настраивать свойства сетки при помощи *Object Inspector* по нажатию клавиши *F11*.
- 11) Модуль *WxSQLScriptGUI* содержит реализацию класса *TWxSQLScriptGUI* для SQL-сценариев для использования в модуле *FrmSQLEx*.

10.3 Модули без GUI для взаимодействия с различными СУБД

- 1) Модуль *WxDBHelp* – служит для создания вспомогательной информационной базы для базы данных выбранной СУБД. Является вспомогательным для модулей: *FrmSQLEx*, *FrmDbIndexes*, *FrmDbOptions*, *FrmDbTables*, *FrmDbTriggers*, *FrmDbProcedures*.
- 2) Модуль *WxDBHelpASA* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *SYBASE ASA*.
- 3) Модуль *WxDBHelpASE* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *SYBASE ASE*.
- 4) Модуль *WxDBHelpDB2* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *DB2*.
- 5) Модуль *WxDBHelpEDB* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *ElevateDB*.
- 6) Модуль *WxDBHelpFirebird* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *Firebird*.
- 7) Модуль *WxDBHelpIFX* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *Informix*.
- 8) Модуль *WxDBHelpInterbase* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *Interbase*.
- 9) Модуль *WxDBHelpIQ* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *Sybase IQ*.
- 10) Модуль *WxDBHelpLinter* – реализация вспомогательной информационной базы для базы данных под управлением российской СУБД *ЛИНТЕР* от компании РЕЛЭКС.
- 11) Модуль *WxDBHelpMSSQL* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *Microsoft SQL SERVER*.
- 12) Модуль *WxDBHelpMySQL* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *MySQL*.
- 13) Модуль *WxDBHelpORACLE* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *ORACLE*.

- 14) Модуль *WxDBHelpPG* – реализация вспомогательной информационной базы для базы данных под управлением СУБД *PostgreSQL*.
- 15) Модуль *WxDBSQLBackUp* предоставляет функциональность по резервному копированию данных.
- 16) Модуль *WxIBAdmin* содержит реализацию функциональности по резервному копированию для СУБД *Interbase*, *Firebird* и «РЕД БАЗА ДАННЫХ». Является вспомогательным для модулей *WxDBSQLBackUp* и *WxDBSQLBackUpFrm*.
- 17) Модуль *WxSQLScript* содержит реализацию класса *TWxSQLScript* для SQL-сценариев для использования в модуле *FrmSQLEx* (через промежуточный класс *TWxSQLScriptGUI*, определённый в модуле *WxSQLScriptGUI*).
- 18) Модуль *WxSQLText* является вспомогательным для модуля *WxSQLScript*. Содержит реализацию используемого в нём класса *TWxSQLText*.
- 19) Модуль *WxTypes* содержит набор констант и типов для выбора СУБД, с которыми будет осуществляться работа.






10.4 *FrmSQLEx* и встроенная среда разработки на языке SQL

Вот как выглядит форма с анализатором запросов:






table_schema	table_name
dbo	DictCity
dbo	Hotels
dbo	WxlAppSettings
dbo	WxlAppUserGroups
dbo	WxlAppUsers
dbo	WxlAutoID
dbo	WxlRecLockList

00,022 [RecordCount 7]
14 912 B

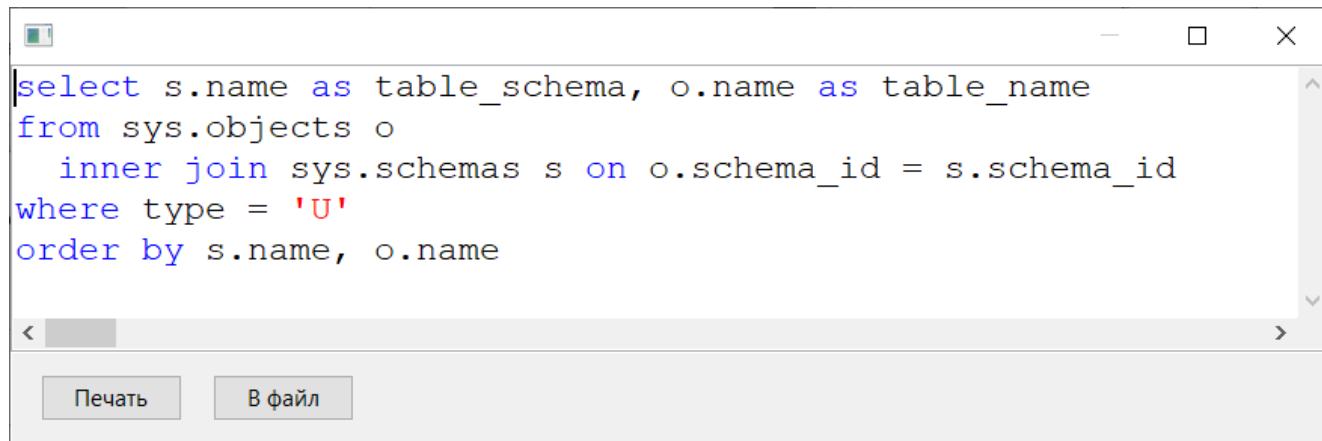
Локализованные названия полей
 Показывать каждый пакет SQL

Как видно из рисунка, в верхней части формы пишется запрос, а в нижней части формы отображается результирующий набор данных. Для выполнения запроса необходимо нажать клавишу «F5», либо кнопку . Кнопка  предназначена для загрузки запроса из файла с расширением «.sql». Кнопка  предназначена для сохранения запроса в файл с расширением «.sql». По соглашению, рекомендуется хранить такие файлы в директории *USERS* созданного типового WXL-приложения. Кнопка  позволяет менять шрифт, которым пишется запрос, путём вызова стандартного диалога пользователя по изменению шрифта. Кнопка  предназначена для распечатки текста. После кнопок анализатора содержится информация о времени выполнения запроса и числе затронутых записей:

00,022 [RecordCount 7]
14 912 B

. Правая часть панели содержит кнопки для использования информационной базы для выбранной базы данных. Кнопка  предназначена для просмотра опций базы данных. Кнопка  вызывает форму, предназначенную для просмотра списков имён таблиц. Путём нажатия правой кнопки мыши можно вызвать дополнительную функциональность по работе с выбранной таблицей. Кнопка  вызывает форму, предназначенную для просмотра списков имён представлений. Путём нажатия правой кнопки мыши можно вызвать дополнительную функциональность по работе с выбранным представлением. Кнопка  вызывает форму, предназначенную для просмотра списков хранимых на сервере процедур. Путём нажатия правой кнопки мыши можно вызвать дополнительную функциональность по работе с хранимыми на сервере процедурами. Кнопка  вызывает форму, предназначенную для просмотра списков хранимых на сервере функций. Путём нажатия правой кнопки мыши можно вызвать дополнительную функциональность по работе с хранимыми на сервере функциями. Галочка «Локализованные названия полей» используется для учёта используемых в приложении дескрипторов полей. В частности, для ранее

рассмотренного примера запрос «select * from DictCity» вернёт набор данных с заголовками полей «ID», «Населённый пункт», «Актуально (0/1)». Анализатор запросов позволяет использовать ключевое слово «GO» для разделения отдельных пакетов. Если дополнительно поставить галочку «Показывать каждый пакет SQL», то каждый пакет будет отображён в отдельном окне.



```
select s.name as table_schema, o.name as table_name
from sys.objects o
    inner join sys.schemas s on o.schema_id = s.schema_id
where type = 'U'
order by s.name, o.name
```

Печать В файл

Кнопка «Печать» предназначена для распечатки запроса, кнопка «В файл» для его сохранения в файл.

10.5 Сохранение и восстановление БД

10.5.1 Общие подходы

В случае возникновения аварийной ситуации данные должны быть защищены от потери и разрушения. Для большинства баз данных СУБД типа клиент-сервер существует соответствующий журнал (протокол) транзакций – файл, куда SQL-сервер записывает все выполняемые транзакции. Существует три основных типа ведения журнала: *undo* («отмена»), *redo* («повторение») и *undo/redo* («отмена/повторение»). Для возобновления информации после системного сбоя необходимо использовать данные протоколов всех разновидностей. Существует механизм сокращения актуальных данных журнала. Он называется методом контрольных точек (*checkpoints*). Кроме того, для сохранения и восстановления баз данных необходимо вести архивы (*archives*). При наличии свежего архива и уцелевшей информации журнала можно восстановить содержимое базы данных даже после полной утраты информации.

Внештатные ситуации и способы выхода из них

1) Ошибочные элементы данных

Наиболее типичной ошибкой является ввод пользователем неправильной информации в базу данных. Для предотвращения подобных ошибок используются внешние ограничения: первичные и внешние ключи, а главное – триггеры. Триггеры представляют собой реакции на различные изменения в базе данных. К примеру, если номер удостоверения состоит из восьми цифр, а введено шесть, триггер позволит обнаружить данную ошибку и устранить её последствия.

2) Разрушение носителя

Если разрушена лишь часть носителя, затрагивающая несколько битов, то данная проблема разрешается при помощи метода контрольных сумм (*Checksums*).

Если повреждения серьёзные (испортилась головка дискового устройства), то для восстановления данных необходимо одно из следующих условий:

2.1) Использовалась схема RAID (Redundant Array of Independent Discs). В этом случае, при выходе из строя одного из дисков, его данные можно восстановить по данным остальных дисков.

2.2) Использование архива, хранящегося на других носителях. Для целесообразности данного метода архивы должны создаваться периодически. Возможно полное или частичное сохранение данных в архивах.

2.3) Сохранение копий (реплик) базы данных в разных местах. Данные копии должны быть согласованы на основании систем репликации данных.

3) Катастрофа

Предположим, что диск с данными базы данных утерян безвозвратно в силу причин, не связанных непосредственно с ним (пожар в здании, диск украден). В этом случае для восстановления данных можно использовать лишь два последних пункта предыдущего раздела.

4) Сбой системы

Чаще всего данная внештатная ситуация возникает из-за недостатков конкретного программного обеспечения. Другой причиной возникновения проблемы являются перебои в электроснабжении. В любом случае, данная

проблема затрагивает транзакции. В результате подобного сбоя становится неясно, выполнена ли та или иная транзакция, поэтому метод повтора транзакций неприменим. Метод решения данной проблемы – ведение подробного журнала транзакций.

10.5.2 Особенности, накладываемые системами репликации

Общая информация

Репликация – это процесс распространения информации между двумя базами данных с целью обеспечения синхронности данных в них. База данных, откуда идёт информация, называется источником, а база данных, куда идёт информация, – приёмником. Следует иметь в виду, что приёмников и источников может быть несколько.

В настоящее время разработчики распределённых систем всё чаще используют для репликации так называемые **технологии репликации и синхронизации данных**. В дальнейшем, именно такие технологии мы и будем называть репликацией. Соответственно, **основная идея репликации – полностью вывести из состава приложений баз данных программный код, реализующий обмен информацией, переложив эту работу на специализированное программное обеспечение (системы репликации)**.

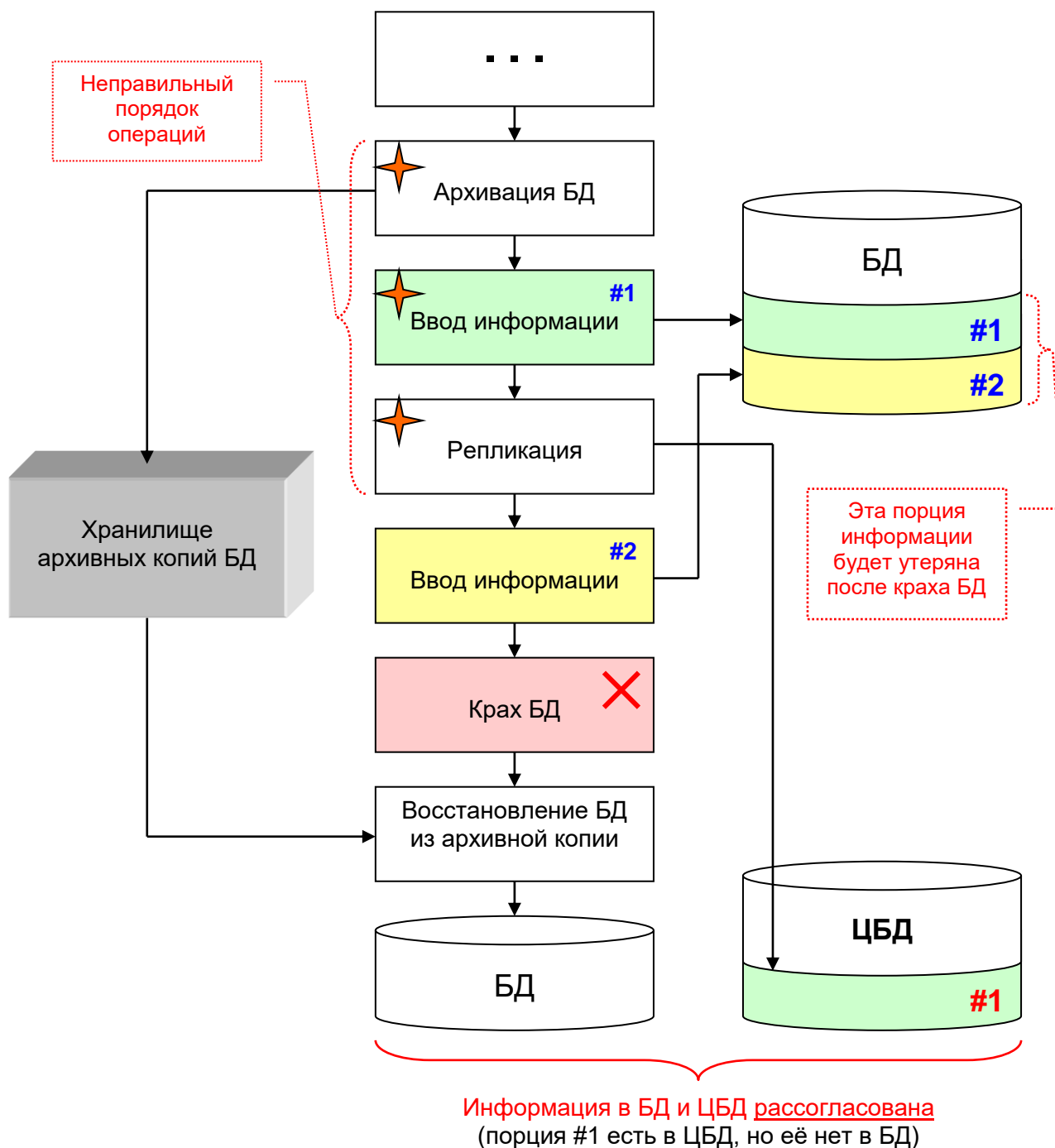
Сформулируем основные понятия, связанные с использованием технологий репликации и синхронизации данных:

Репликация – это процесс распространения информации между несколькими базами данных, выполняемый неким специализированным программным обеспечением.

Распределённая информационная система – это информационная система, отдельные компоненты которой (взаимодействующие субъекты и их базы данных) размещены в территориально удалённых друг от друга точках.

Консолидирующая база данных – это база данных, в которую стекается информация из баз данных других субъектов распределённой системы и из которой информация распространяется (тиражируется) в базы данных других субъектов системы.

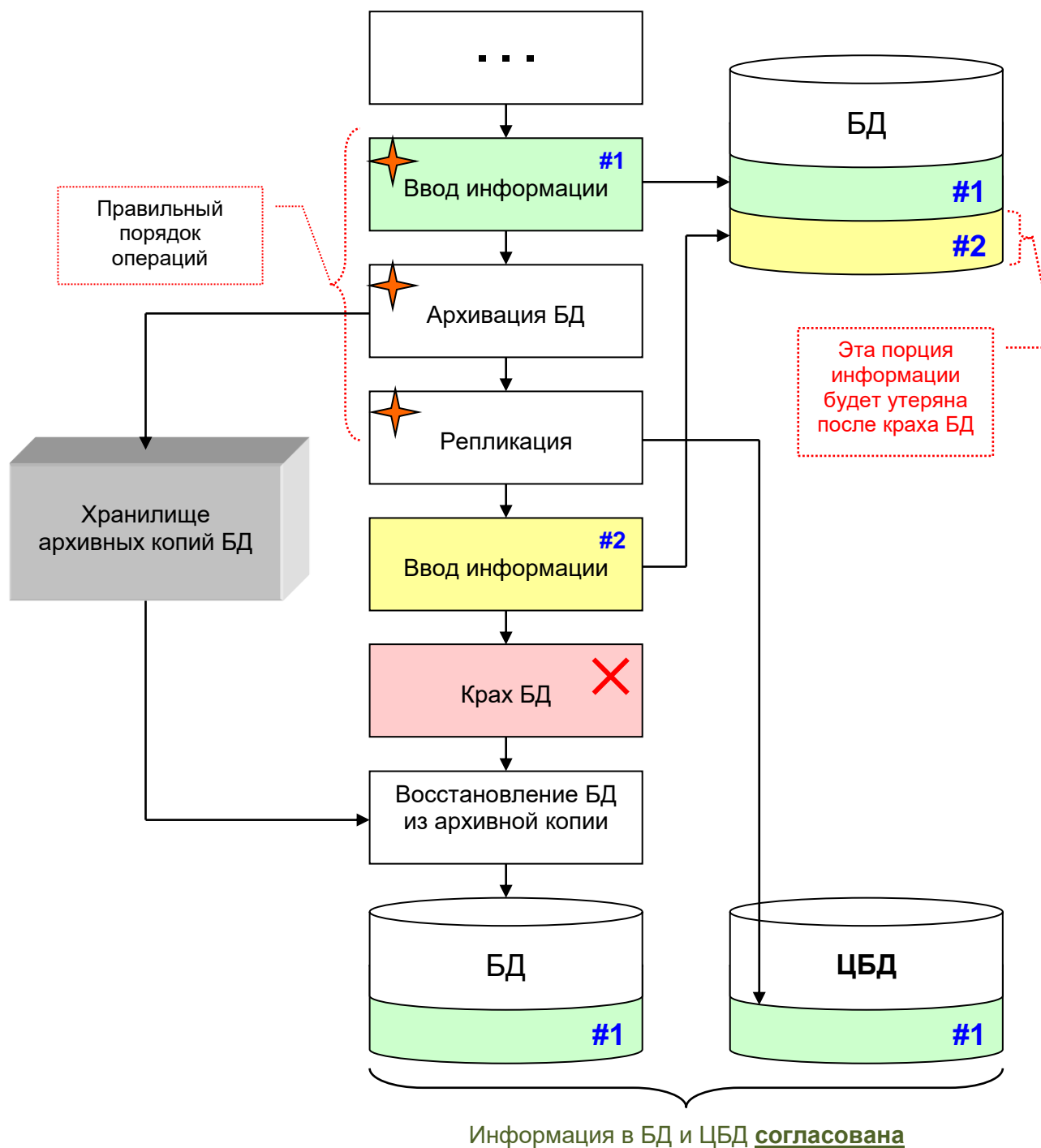
Приведём пример неправильного использования репликации. Пусть после очередного резервного копирования клиентской базы данных данные были модифицированы, а затем и реплицированы. Пусть в этот момент произойдёт крах системы. В этом случае произойдёт откат к ближайшей резервной копии базы данных.



Неправильная технология архивации БД в распределенной системе

Таким образом, результаты модификации данных будут потеряны в клиентской БД, но будут занесены в консолидирующую базу данных. Следовательно, будет нарушена согласованность информации во всей системе, а

сама распределённая информационная система будет находиться в состоянии полного краха. Причина таких разрушительных последствий заключается в том, что репликация была произведена над данными, для которых не было создано резервной копии. Таким образом, решение проблемы заключается в резервном копировании данных перед каждой репликацией. Перестроим схему, так, чтобы крах клиентской БД не был губителен для распределённой системы в целом.



Правильная технология архивации БД в распределенной системе

Видно, что в этом случае, после краха клиентской БД, согласованность информации в распределённой системе не нарушена. Таким образом, для эффективного использования систем репликации с точки зрения сохранения и восстановления данных необходимо ввести **запрет на тиражирование информации, для которой не было создано резервной копии**. Аналогично, если выполняется операция сохранения журнала транзакций, то репликация должна охватывать только то подмножество транзакций, для которого была выполнена операция резервного копирования.

10.5.3 Средства выполнения *Backup* в *WXL*-приложениях

На момент составления документации библиотека *WXL* позволяла поддерживать возможности *BackUp* для следующих СУБД: *Microsoft SQL Server*, *Sybase ASA*, *Sybase ASE*, *Sybase IQ*, *Lintar*, *Interbase* и *Firebird*. Для выполнения резервного копирования с целью дальнейшего восстановления данных применяется модуль *WxDBSQLBackUp*. Данный модуль предоставляет две возможности для резервного копирования данных: резервное копирование базы данных в указанное место на диске, резервное копирование журнала транзакций в указанное место на диске (кроме *Interbase* и *Firebird*). Резервное копирование может проходить с использованием *RSC*-процедур или без их использования.

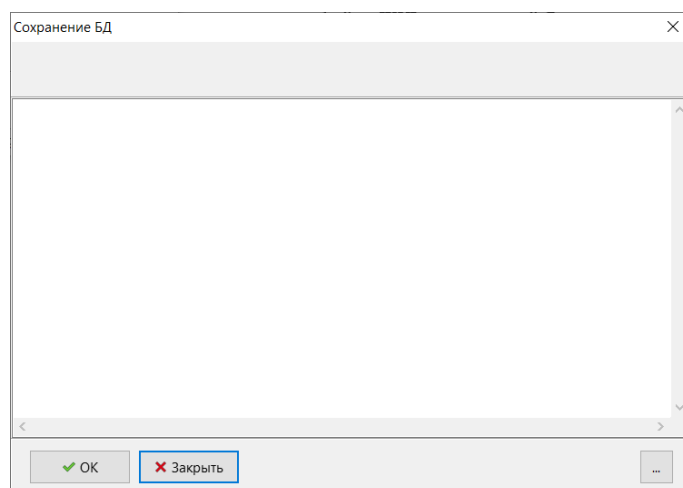
В случае, если *RSC*-процедуры не установлены, то для резервного копирования журнала транзакций *Microsoft SQL Server* необходимо, чтобы *Recovery Model* не была *Simple*. Изменить *Recovery Model* в базе данных под управлением *Microsoft SQL Server* можно, например, следующим образом: войти в *Management Studio*, нажать правой кнопкой мыши на требуемую базу данных, «Свойства», «Параметры», «Модель восстановления» («*Recovery Model*»). Для *Sybase ASA* в зависимости от используемой версии может использоваться либо *SQL*-команда *BACKUP DATABASE*, либо внешняя утилита *dbbackup*. Для *Interbase* и *Firebird* функциональность по резервному копированию вынесена в модуль *WxIBAdmin* и использует вызов сервиса *service_mgr*.

В случае, если *RSC*-процедуры установлены, то для выполнения резервного копирования используются процедуры *RSC_BACKUPDATABASE* (резервное копирование всей базы данных) и *RSC_BACKUPLOG* (резервное копирование

журнала транзакций), имеющие один строковый параметр – имя директории, в которой будут размещены резервные копии данных. После первой попытки проведения сеанса обмена в клиентской базе данных при использовании *rsxAgent* автоматически создаются обе указанные хранимые на сервере процедуры, предназначенные для выполнения операции *BACKUP*, а также вспомогательные процедуры *RSC_BackupBefore* и *RSC_BackupAfter*. Кроме выполнения операции *BACKUP*, указанные процедуры *rsxAgent*, используя внутренние процедуры *RSC_BackupBefore* и *RSC_BackupAfter*, записывают в специальную таблицу *RSC_BackupTbl* информацию о текущих значениях степени свежести записей в каждой экспортируемой таблице к моменту выполнения *Backup*. Это позволяет агенту синхронизации в каждом сеансе определять, для каких записей реплицируемых таблиц был выполнен *Backup*. Эти записи могут участвовать в репликации. В любом случае, модуль *WxDbSQLBackUp* позволяет получить историю сделанных резервных копирований. Для *Microsoft SQL Server* эти данные извлекаются из базы данных *msdb*. Для *Sybase ASA* и *Sybase IQ* используется процедура *sa_read_backup_history*. Для *Interbase* используется таблица *RDB\$JOURNAL_ARCHIVES*. Для *Firebird* используется таблица *RDB\$BACKUP_HISTORY*.

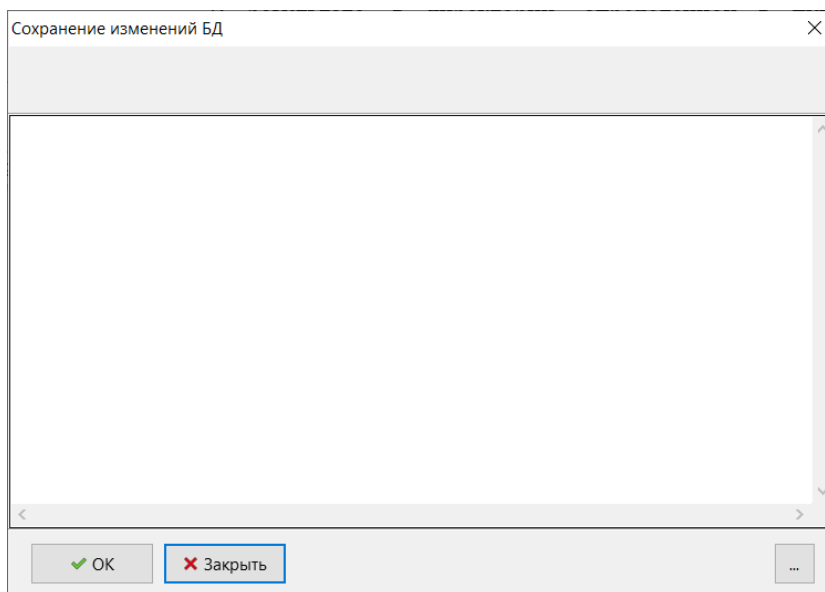
В типовом *WXL*-приложении для осуществления резервного копирования базы данных и журнала транзакций используются пункты подменю «АДМИНИСТРАТОР» «Сохранение БД» и «Сохранение изменений БД».

Пункт «Сохранение БД» вызывает форму, предназначенную для сохранения БД, реализованную в модуле *WxDbSQLBackUp*:



Кнопка «Заккрыть» отменяет выбор данного пункта меню, кнопка «ОК» проводит сохранение базы данных в выбранную в пункте меню «Настройки» директорию, кнопка в правой нижней части формы позволяет посмотреть историю резервных копирований. В результате в директории, определённой в пункте меню «Настройки», появляется резервная копия базы данных с именем «ИМЯБАЗЫДААННЫХ_DAT.BCK» («*TOURISM_DAT.BCK*»).

Пункт меню «Сохранение изменений БД» вызывает форму, предназначенную для сохранения изменений БД, реализованную в модуле *WxDbSQLBackUp*:



Кнопка «Заккрыть» отменяет выбор данного пункта меню, кнопка «ОК» проводит сохранение журнала транзакций базы данных в выбранную в пункте меню «Настройки» директорию, кнопка в правой нижней части формы позволяет посмотреть историю резервных копирований. В результате в директории, определённой в пункте меню «Настройки», появляется резервная копия журнала транзакций базы данных с именем «ИМЯБАЗЫДААННЫХ_LOG.BCK» («*TOURISM_LOG.BCK*»).

10.6 Генерация уникальных идентификаторов

Иногда необходимо, чтобы в таблице базы данных был столбец, значение которого уникально идентифицирует всю запись в целом. Существует две наиболее часто встречаемые ситуации, когда это необходимо. Первая ситуация – это использование первичных и внешних ключей для поддержания ссылочной целостности базы данных. Вторая ситуация – это использование уникальных

значений полей для избегания конфликтов нескольких пользователей, одновременно изменяющих одну и ту же запись. Для решения первой задачи чаще всего используются так называемые автоинкрементные поля. Для решения второй задачи чаще всего используются поля, называемые большинством СУБД *rowversion*. Если используются автоинкрементные поля, то первой записи в таблице ставится в соответствие определённое значение поля (например, 0 или 1) и задаётся приращение. В этом случае вторая запись будет иметь значение автоинкрементного столбца, представляющее собой начальное значение плюс приращение. Третья запись будет иметь значение автоинкрементного поля на приращение больше, чем значение автоинкрементного столбца второй записи и так далее. Возможна ситуация, когда используется вычитание и большие значения в качестве начальных значений. *Rowversion*-поля всегда имеют уникальные значения в пределах таблицы базы данных. Их значения изменяются после каждой операции вставки или изменения записи. В результате, если один пользователь изменил значения какого-либо поля записи, то другой уже не сможет сделать этого, так как изменилось значение у поля типа *rowversion*. Для поддержания данного механизма необходимо задавать *SQL*-запросы, использующие *where* для поля типа *rowversion*. Данная особенность полей типа *rowversion* делает их непригодными для использования в качестве первичных и внешних ключей.

Давайте рассмотрим присутствие или отсутствие полей такого рода среди наиболее распространённых СУБД.

<i>СУБД</i>	<i>Rowversion</i>	<i>Autoincrement</i>
<i>Microsoft SQL Server</i>	<i>Timestamp</i> или <i>Rowversion</i>	<i>identity</i>
<i>ORACLE</i>	Близкая по функциональности псевдоколонка <i>ORA_ROWSCN</i> .	Начиная с версии 12с, <i>GENERATED ALWAYS as IDENTITY (START with 1 INCREMENT by 1)</i>
<i>Sybase ASE</i>	<i>Timestamp</i>	<i>identity</i>
<i>Sybase ASA</i>		<i>identity</i>
<i>MySQL</i>		<i>auto_increment</i>
<i>PostgreSQL</i>		<i>smallserial, serial</i> и <i>bigserial</i> столбцы
<i>Lintar</i>		<i>autoinc</i>
<i>Firebird</i>		Начиная с версии 3, <i>generated by default as identity</i>
<i>Informix</i>		<i>Serial</i> или <i>serial8</i> столбцы

DB2	Близкие по функциональности выражения <i>ROW CHANGE TIMESTAMP</i> и <i>ROW CHANGE TOKEN</i>	<i>GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1)</i>
-----	---	---

В *Microsoft SQL Server* существует тип данных *timestamp*. Причём его реализация отличается от реализаций типа *timestamp* в других *SQL*-серверах. В *Microsoft SQL Server* этот тип правильнее было бы именовать «*RowVersion*». Именно по этой причине в *Microsoft SQL Server 2008* введён тип *RowVersion*, который является синонимом типа *timestamp*. Поведение *timestamp* в *Microsoft SQL Server* базируется на глобальной переменной базы данных @@DBTS (*DataBase TimeStamp*). Фактически @@DBTS, будучи преобразованным в 8-байтовое целое, представляет собой глобальный автоинкрементальный счётчик, значение которого монотонно увеличивается на 1 при добавлении/исправлении любой записи любой таблицы, содержащей поле *timestamp* (само поле при этом принимает текущее значение @@DBTS). Преобразование полей *timestamp* в 8-байтовое целое в *Microsoft SQL Server* осуществляется так: `cast(<поле> as bigint)`. Максимальное значение полей *bigint* $2^{63}-1$ (9,223,372,036,854,775,807). Этого более чем достаточно практически для любой прикладной задачи, использующей механизм *timestamp* для генерации *RowVersion* записей.

Следует иметь в виду, что отсутствие predefined типа не означает отсутствие механизма подобных полей. Для получения идентичной функциональности может использоваться механизм *sequence (generator)+ trigger*.

В библиотеке *WXL* при создании типового *WXL*-приложения при помощи генератора *wxlgen* можно использовать функциональность по генерации собственных уникальных идентификаторов. Приведем пример *SQL*-скриптов, необходимых для создания объектов по поддержке генерации уникальных идентификаторов для *Microsoft SQL Server*:

```

/*
if isnull(objectproperty(object_id(N'[dbo].[WxlAutoID]'), N'IsTable'), -1) = 1
    drop table [dbo].[WxlAutoID]
go
*/

create table dbo.WxlAutoID(
    FieldName nvarchar(128) not null,
    ID int not null,
    constraint PK_WxlAutoID primary key (FieldName)

```

```

go

/*
if isnull(objectproperty(object_id(N'[dbo].[WxlGetNextID]'), N'IsProcedure'),-1)=1
    drop procedure [dbo].[WxlGetNextID]
go
*/

create procedure dbo.WxlGetNextID(
    @FieldName nvarchar(128),
    @Ret      int out)
as
begin
    set nocount on;

    declare @StartID int;
    declare @ID int;

    set @StartID = 0;

    begin tran WxlSetNextID;

    update dbo.WxlAutoID set @ID = ID = ID + 1 where FieldName = @FieldName;
    if @@rowcount = 0
    begin
        insert into dbo.WxlAutoID (FieldName, ID) values (@FieldName, 1);
        set @ID = 1;
    end;

    commit tran WxlSetNextID;

    set @Ret = @StartID + @ID;
    return @Ret;
end;
go

/*=== TEST
declare @Ret int;
exec dbo.WxlGetNextID 'FieldName', @Ret out;
select @Ret as Result;
*/

```

В типовом WXL-приложении функция генератора нового идентификатора помещается в модуль «ПРИЛОЖЕНИЕДАТАМАН» («*travelsdataman*») и называется *GetNextID*. Данная функция, в свою очередь, используется в модулях с именами вида «DlгИМЯТАБЛИЦЫ» («*dlghotels*»). Например, при обработке нажатия на кнопку «ОК» можно записать следующий код, если диалог осуществляет вставку записей в таблицу (*daInsert*, *daInsertCycle*):

```

FData.MyInteger := GetNextId(fnMyInteger);
FDataSet.RecordInsert(FData, FDataSet.GetDataTitle(FData));
GDataPrev := FData;
Inc(FCount);
ModalResult := mrOk;

```

При этом в текст процедуры *FormCreate* необходимо добавить следующий код:
WxEntryMyInteger.Enabled := false;

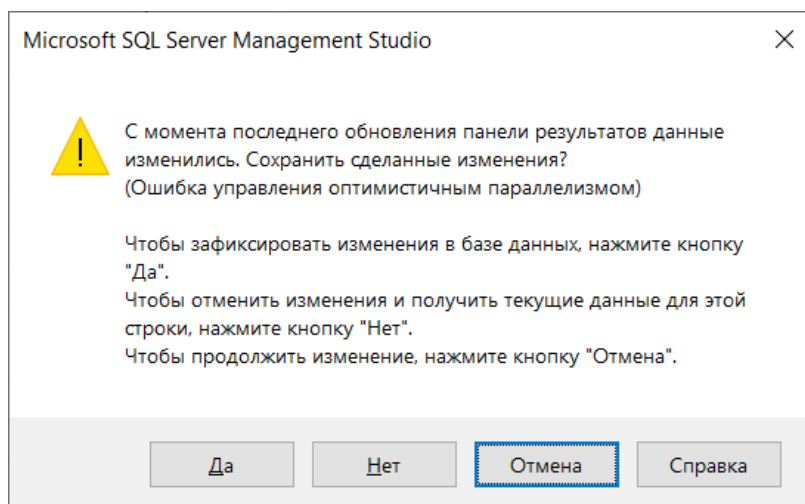
```
WxEntryMyInteger.EntryOptions := [eoBlankEnable];
```

В результате при каждой вставке записи в таблицу поле *MyInteger* будет заполняться последовательными целыми числами. При необходимости использования более сложных уникальных идентификаторов необходимо заменить хранимую на сервере процедуру *WxlGetNextId*.

10.7 Блокировка записей в таблицах БД

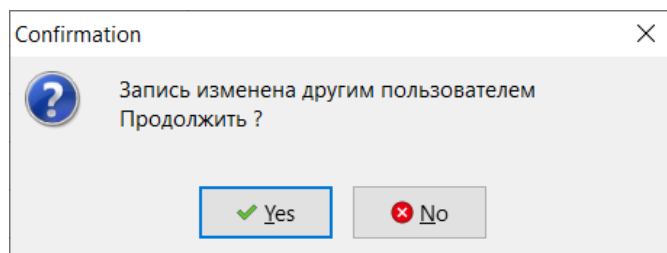
При работе с одной и той же базой данных часто возникает конфликт пользователей, которые хотят изменить одни и те же записи определённой таблицы. Существует две политики избегания подобной ситуации. Большинство SQL-серверов осуществляет так называемую «оптимистическую блокировку пользователей». То есть вне зависимости от того, работают ли другие пользователи с заданной таблицей, каждый следующий обращающийся получает к ней доступ, так как считается, что в большинстве случаев происходит обращения только для считывания записей.

Пусть в приложении баз данных была изменена запись таблицы базы данных *Microsoft SQL Server*, а другой пользователь хочет изменить ту же самую запись при помощи *Management Studio*. При сохранении изменений второй пользователь получает следующее сообщение:



Соответственно, выбрав «Да», он уничтожит изменения, сделанные первым пользователем. Если он выберет «Нет», то уничтожит изменения, которые сделал сам. Можно, конечно, нажать и «Отмена», но для выделения тех записей, которые трогать нельзя, уйдёт достаточно много времени.

Пусть мы столкнулись с обратной ситуацией. Тогда, пользуясь *WXL*-приложением, при попытке изменения записи, мы получаем сообщение:

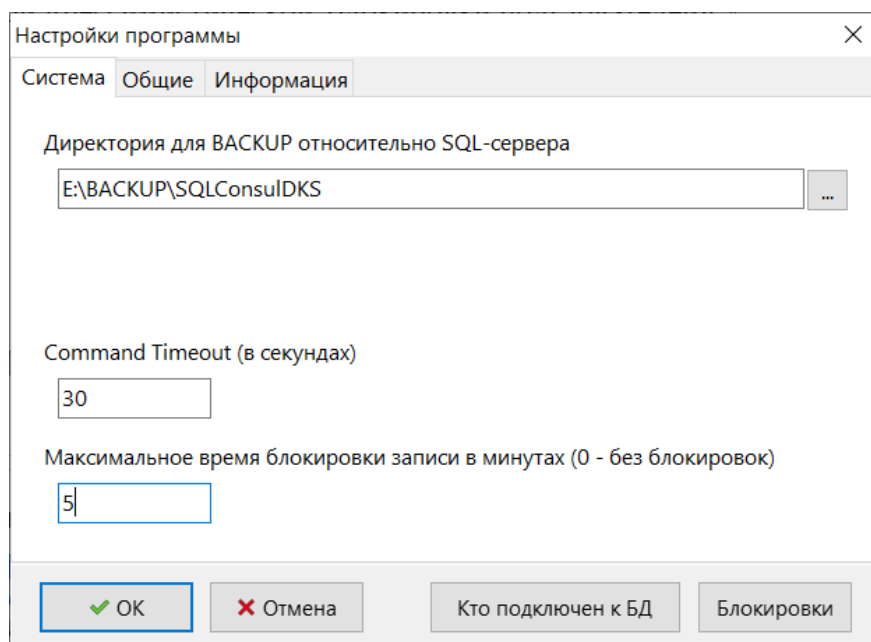


Соответственно, если проигнорировать это сообщение и нажать «*Yes*», то будет загружена новая, уже изменённая запись, в которую можно вносить изменения.

Таким образом, мы видим, что оптимистическая блокировка пользователей приводит к потере работы пользователя, связанной с изменениями, сделанными в строках таблиц.

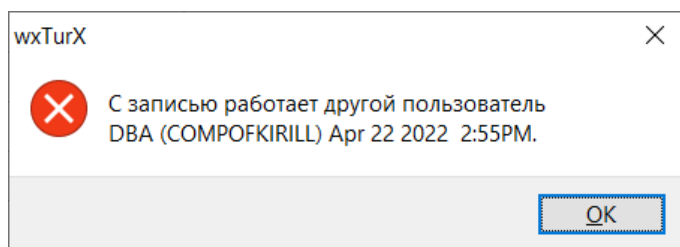
В библиотеке *WXL* при создании типового *WX*-приложения при помощи генератора *WxlGen* можно включить функциональность по обеспечению так называемой «пессимистической блокировки пользователей».

Для задания пессимистической блокировки пользователей необходимо выбрать пункт «Настройки программы» подменю «Администратор». В появившейся форме необходимо установить любое максимальное время блокировки, отличное от нуля.

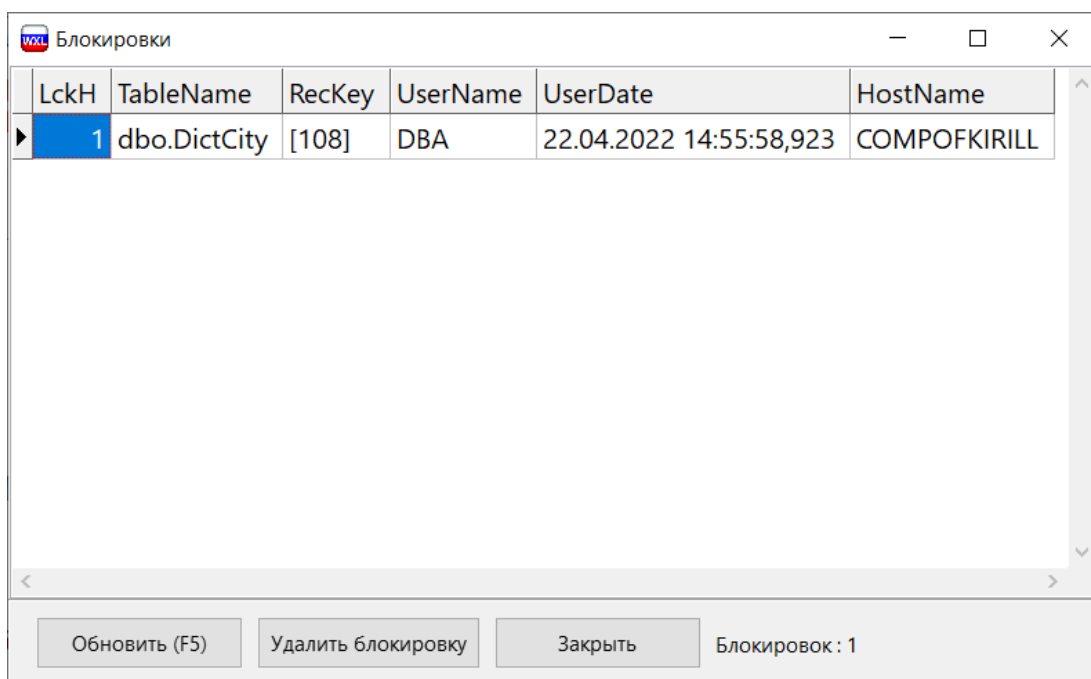


Для поддержки функциональности по пессимистической блокировке в модуле «ПРИЛОЖЕНИЕDataMan» («*travelsdataman*») процедура «*EnableRecordLock*» с одним параметром *Boolean*, который необходимо присвоить значению *True*, если следует использовать данную функциональность, и *False*, если не следует. В модуле *DlgAppSettings* в процедуре *ApplyAppSettings* она вызывается следующим образом: `WXTURDATAMAN.EnableRecordLock(AData.RecLockTimeout > 0);`

В результате, как только один пользователь начинает работать с набором данных при использовании блокировок, для других пользователей автоматически отключается возможность редактировать те записи, которые в данный момент редактирует данный пользователь. Вот как выглядит исключительная ситуация, возникающая, если пользователь WXL-приложения пытается редактировать заблокированную запись:



В типовом WXL-приложении администратор может удалить блокировку, выбрав пункт «Настройки программы» подменю «Администратор» и нажав кнопку блокировки. В результате появится таблица блокировок:



LckH	TableName	RecKey	UserName	UserDate	HostName
1	dbo.DictCity	[108]	DBA	22.04.2022 14:55:58,923	COMPOFKIRILL

Нажав на кнопку «Удалить блокировку», можно снять блокировку с заданных записей.

Приведем пример *SQL*-скриптов, создаваемых генератором *wxlGen*, которые необходимы для создания объектов по пессимистической блокировке записей в таблицах базы данных под управлением *Microsoft SQL Server*:

```
/*
if isnull(objectproperty(object_id(N'[dbo].[WxlRecLockList]'), N'IsTable'), -1) = 1
    drop table [dbo].[WxlRecLockList]
go
*/

create table dbo.WxlRecLockList(
    LckH          int          identity,
    TableName     nvarchar(128) not null,
    RecKey        nvarchar(255) not null,
    UserName      nvarchar(20)  not null,
    UserDate      datetime      not null,
    HostName      nvarchar(20)   null,
    constraint PK_WxlRecLockList primary key (LckH))
go

create unique index IX_WxlRecLockList on dbo.WxlRecLockList (TableName, RecKey)
go

/*
if isnull(objectproperty(object_id(N'[dbo].[WxlRecUnlock]'), N'IsProcedure'), -1) = 1
    drop procedure [dbo].[WxlRecUnlock]
go
*/

create procedure dbo.WxlRecUnlock(
    @LckH int,
    @Ret  int out)
as
begin
    set nocount on;

    set @Ret = -888;
    delete from dbo.WxlRecLockList where LckH = @LckH;
    set @Ret = @@rowcount;
    return @Ret;
end;
go

/*
if isnull(objectproperty(object_id(N'[dbo].[WxlRecLock]'), N'IsProcedure'), -1) = 1
    drop procedure [dbo].[WxlRecLock]
go
*/

create procedure dbo.WxlRecLock(
    @TableName  nvarchar(128),
    @RecKey     nvarchar(255),
    @UserName   nvarchar(20),
    @HostName   nvarchar(20),
    @RecLockTimeout int,
    @Ret        int out, /* @LckH */
    @Msg        nvarchar(128) out)
as
```

```

begin
    set nocount on;
    set @Ret = -888;
    set @Msg = '';
    declare
        @UserName nvarchar(20),
        @UserDate datetime,
        @HostName nvarchar(20),
        @xRet int,
        @xLckH int;

    set @xLckH = (select LckH from dbo.WxlRecLockList
    where TableName = @TableName
    and RecKey = @RecKey);
    set @xLckH = isnull(@xLckH, -1);

    if @xLckH > -1
    begin
        set @xUserDate = (select UserDate from dbo.WxlRecLockList where LckH = @xLckH);
        if datediff(minute, @xUserDate, GetDate()) >= @RecLockTimeout
        begin
            exec dbo.WxlRecUnlock @xLckH, @xRet out;
        end
        else
        begin
            set @UserName = (select UserName from dbo.WxlRecLockList where LckH = @xLckH);
            set @HostName = (select HostName from dbo.WxlRecLockList where LckH = @xLckH);
            set @Msg = @UserName + N' (' + @HostName + N') ' + convert(nvarchar,
@xUserDate);
            return @Ret;
        end;
    end;

    insert into dbo.WxlRecLockList(TableName, RecKey, UserName, UserDate, HostName)
    values (@TableName, @RecKey, @UserName, GetDate(), @HostName);
    if @@error <> 0 return @Ret;

    set @Ret = (select LckH from dbo.WxlRecLockList
    where TableName = @TableName
    and RecKey = @RecKey);
    return @Ret;
end;
go

/*=== TEST
declare @Ret int;
declare @Msg nvarchar(128);
exec dbo.WxlRecLock
    'TableName',
    'RecKey',
    'UserName',
    'HostName',
    1,
    @Ret out,
    @Msg out;
select @Ret as Result, @Msg as Msg;

declare @Ret int;
exec dbo.WxlRecUnLock 1, @Ret out;
select @Ret as Result;
*/

```

11. Работа с пользователями

11.1 Многопользовательский доступ к данным

В библиотеке *WXL* акцентировано внимание на создании функциональности по обеспечению дополнительной безопасности при работе с базами данных. Для этого перед началом работы с базой данных необходимо выполнить создание таблиц пользователей и групп пользователей данной базы данных. Приведём *SQL*-скрипты для создания таких таблиц для СУБД *Microsoft SQL Server*:

1) Создание таблицы групп пользователей.

```
/*
if isnull(objectproperty(object_id(N'[dbo].[WxlAppUserGroups]'), N'IsTable'), -1) =
1
drop table [dbo].[WxlAppUserGroups]
go
*/

create table dbo.WxlAppUserGroups (
    GroupName          nvarchar(20) not null,
    ConnectionString  ntext          null,
    Settings           ntext          null,
constraint PK_WxlAppUserGroups primary key (GroupName))
go
```

1.1) Поле таблицы *GroupName* – название группы пользователей.

1.2) Поле таблицы *ConnectionString* – строка соединения.

1.3) Поле таблицы *Settings* – настройки группы пользователей. Это строка специального вида, по которой однозначно восстанавливаются настройки группы пользователей: доступные пункты меню, настройки рабочего места.

2) Создание таблицы пользователей.

```
/*
if isnull(objectproperty(object_id(N'[dbo].[WxlAppUsers]'), N'IsTable'), -1) = 1
drop table [dbo].[WxlAppUsers]
go
*/

create table dbo.WxlAppUsers (
    GroupName          nvarchar(20) not null,
    RegName            nvarchar(20) not null,
    Password           nvarchar(170) null,
    Names              nvarchar(32) null,
    FullName           nvarchar(80) null,
    Post               nvarchar(32) null,
    Phone              nvarchar(32) null,
    PrivateDirW        nvarchar(255) null,
    PrivateDirU        nvarchar(255) null,
    UpDownAsTab        int          null, /* WXENTRY.WxUpDownAsTab: Boolean */
    EnterAsTab         int          null, /* WXENTRY.WxEnterAsTab: Boolean */
    DictSearch         int          null, /* WXENTRY.WxIncrementalDictSearch: Boolean */
    UseEntryActiveColor int          null, /* WXENTRY.WxUseEntryActiveColor: Boolean */
    EntryActiveColor   nvarchar(40) null, /* WXENTRY.WxEntryActiveColor: TColor */
```

```

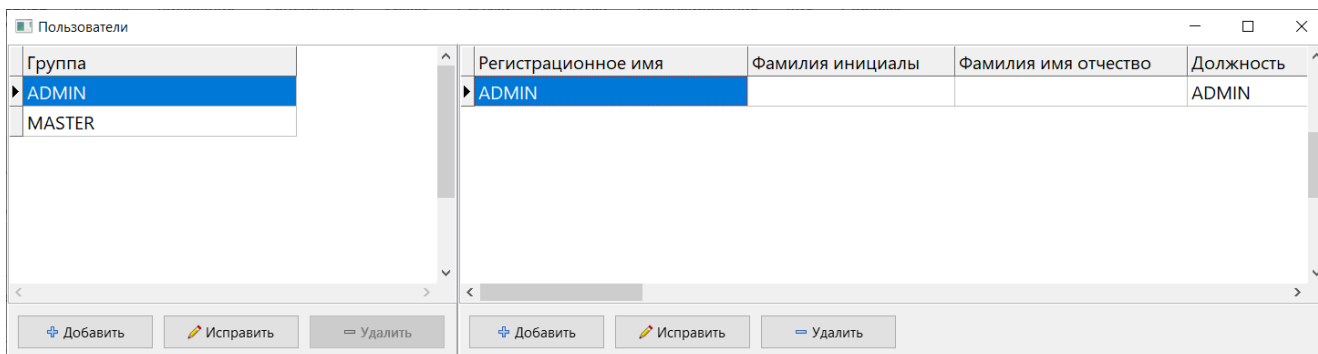
Settings          ntext          null,
constraint PK_WxlAppUsers primary key (GroupName, RegName)
go

create unique index IX_WxlAppUsers_RegName on dbo.WxlAppUsers (RegName)
go

```

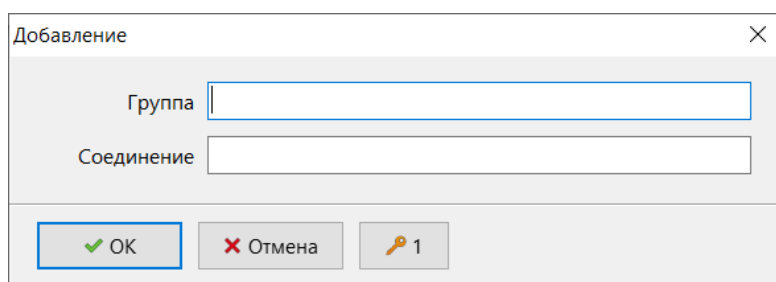
- 2.1) Поле таблицы *GroupName* – название группы пользователей, за которой закреплён данный пользователь.
- 2.2) Поле таблицы *RegName* – регистрационное имя пользователя.
- 2.3) Поле таблицы *Password* – пароль пользователя.
- 2.4) Поле таблицы *Names* – фамилия и инициалы пользователя.
- 2.5) Поле таблицы *FullName* – фамилия, имя и отчество пользователя.
- 2.6) Поле таблицы *Post* – должность пользователя.
- 2.7) Поле таблицы *Phone* – телефонный номер пользователя.
- 2.8) Поле таблицы *PrivateDirW* – личная директория пользователя в операционных системах семейства *Windows*.
- 2.9) Поле таблицы *PrivateDirU* – личная директория пользователя в операционных системах семейства *UNIX*.
- 2.10) Поле таблицы *UpDownAsTab* – используется ли пользователем стрелки для перемещений между компонентами на форме.
- 2.11) Поле таблицы *EnterAsTab* – используется ли пользователем клавиша «*Enter*» для перемещений между компонентами на форме.
- 2.12) Поле таблицы *DictSearch* – используется ли пользователем режим подсказок при вводе, если к компоненту, на который осуществляется ввод, присоединён словарь (справочник).
- 2.13) Поле таблицы *UseEntryActiveColor* – используется пользователем, чтобы определить использовать ли цвет активного поля.
- 2.14) Поле таблицы *EntryActiveColor* – используемый пользователем цвет активного поля.
- 2.15) Поле таблицы *Settings* – настройки пользователя. Строка специального вида, по которой однозначно восстанавливаются настройки пользователя: доступные пункты меню, настройки рабочего места.


В типовом WXL-приложении для создания пользователей и групп пользователей служит пункт «Пользователи» подменю «Администратор» главного меню WXL-приложения. Вот как выглядит всплывающая в этом случае форма для добавки новых групп пользователей и новых пользователей:

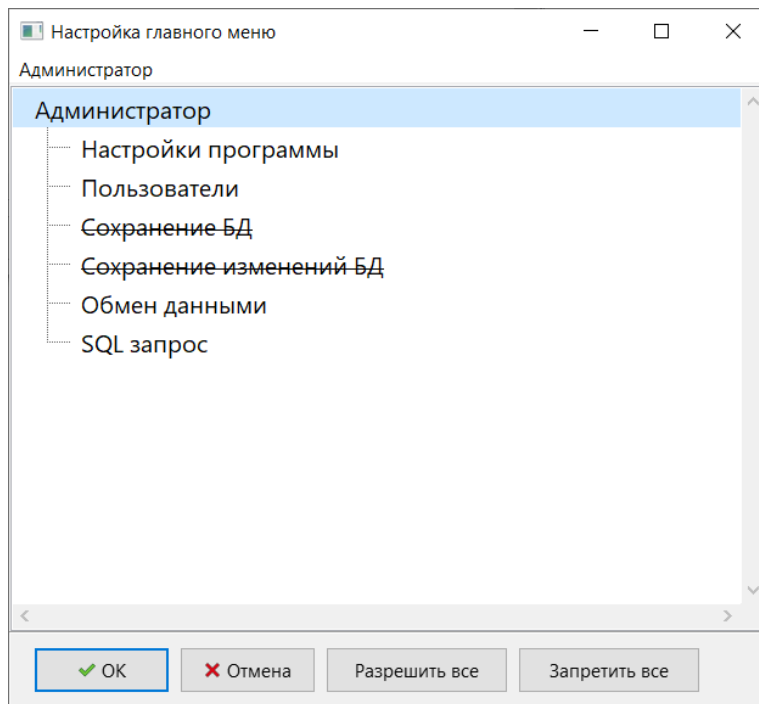


Сама данная форма определена в модуле *WxUserFrm*.

Соответственно, нажимая «Добавить» в левой части формы, можно добавить ещё одну группу пользователей WXL-приложения, а, нажав кнопку «Добавить» в правой части формы, можно добавить пользователя в выделенную группу пользователей слева. Соответственно, кнопки «Исправить» предназначены для редактирования существующих пользователей и групп пользователей WXL-приложения. Кнопки «Удалить» предназначены для удаления выбранной группы пользователей, либо пользователя внутри группы пользователей. Причём следует иметь в виду, что группу пользователей можно удалить в том, и только в том случае, если она не содержит пользователей. Вот как выглядит диалог добавления-редактирования новой группы пользователей:




Нажатие на кнопку  позволяет выбрать доступные/недоступные пункты меню для заданной группы пользователей. Данная функциональность реализована в модуле *WxMenuTune*. Вот как выглядит в общем случае форма настройки меню модулем *WxMenuTune*:



Недоступные пункты меню зачёркнуты, доступные – не зачёркнуты. Кнопка «Разрешить все» делает доступными все пункты меню, кнопка «Запретить все» делает недоступными все пункты меню. Делать пункты меню доступными/недоступными можно при помощи клавиши «*Insert*».

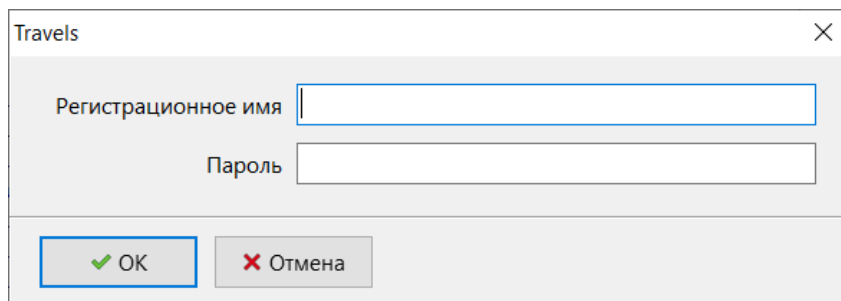
Вот как выглядит диалог добавления-редактирования нового пользователя:



Нажатие на кнопку  позволяет выбрать доступные/недоступные пункты меню для заданного пользователя. Данная функциональность реализована в модуле *WxMenuTune*.

Диалоги для добавления/исправления пользователя или группы пользователей реализованы соответственно в модулях *WxUserDlg* и *WxUserGroupDlg*.

Функциональность по обеспечению защиты доступа к заданному *WXL*-приложению реализована в модуле *WxUserRegDlg*. Чтобы получить доступ к функциональности типового *WXL*-приложения необходимо пройти регистрацию для использования *WXL*-приложения. Форма регистрации имеет следующий вид:



Так как при первом запуске сгенерированного *WXL*-приложения пользователи конкретного *WXL*-приложения неизвестны генератору *wxlGen*, то существуют регистрационное имя пользователя и соответствующий ему пароль, которые можно использовать при первом использовании приложения. Это пользователь *ADMIN* с паролем *ADMIN*. Следует иметь в виду, что данный пользователь или его пароль должен быть изменён при первом запуске *WXL*-приложения, так как, в противном случае, защита доступа к *WXL*-приложению теряет смысл.

В общем случае работа с заданными таблицами пользователей и групп пользователей осуществляется при помощи модуля *WxUserFrm*. Можно заметить, что таблица пользователей содержит поле паролей. Возникает вопрос, нельзя ли разрушить данную функциональность по обеспечению безопасности при помощи обращения через *Management Studio Microsoft SQL Server* или другое аналогичное средство к таблице пользователей и, таким образом, узнать все пароли. Дело в том, что пароли в таблице пользователей хранятся в специально зашифрованном виде и узнать их таким образом не удастся. Из этого сразу же делается вывод, что и заполнять пароли пользователей необходимо только посредством

функциональности модуля *WxUserFrm*, так как в противном случае проверка даст отрицательный результат даже на введённый вами пароль, потому что она осуществляет сверку со значением, подвергнутом обратному преобразованию к зашифровке.

11.2 Модули для работы с пользователями *wxl*-приложений

Все модули для работы с пользователями *wxl*-приложений с графическим интерфейсом, размещены в подпапке *gui* исходников библиотеки *WXL*.

- 1) Модуль *WxUserTbl* является вспомогательным для других модулей работы с пользователями.
- 2) Модуль *WxUserRegDlg* обеспечивает безопасность при доступе к *WXL*-приложениям.
- 3) Модуль *WxUserGroupDlg* содержит форму для добавления новой группы пользователей *WXL*-приложения.
- 4) Модуль *WxUserFrm* является главным модулем для осуществления удаления, редактирования и добавления пользователей *WXL*-приложения и их групп.
- 5) Модуль *WxUserDlg* содержит форму для добавления нового пользователя *WXL*-приложения.

12. Прочие модули библиотеки WXL

- 1) GUI-модули, реализующие другие возможности (размещены в подпапке *gui* исходников библиотеки WXL).
 - 1.1) Модуль *WxAppUtils* содержит необходимые библиотеке WXL формы. Также в данном модуле содержатся функции для работы с экранными подсказками, функции вызова сообщений с ошибками, предупреждениями, информацией, запросами на подтверждение.
 - 1.2) Модуль *WxBitMask* содержит класс, ориентированный на удобную работу с *CheckListBox*.
 - 1.3) Модуль *WxCalendar* предназначен для работы с календарём.
 - 1.4) Модуль *WxDlgScale* содержит реализацию диалога *TWxDialogScale* для настройки масштабирования форм в WXL-приложениях.
 - 1.5) Модуль *WxInpStr* содержит реализацию диалога для ввода строки (с поддержкой различных строковых типов данных).
 - 1.6) Модуль *WxLogDbGrid* содержит реализацию лога на основе сетки *WxGrid*.
 - 1.7) Модуль *WxLogSynEdit* содержит реализацию лога с подсветкой синтаксиса на основе *TWxSynEdit*.
 - 1.8) Модуль *WxLogVST* содержит реализацию лога на основе *TVirtualStringTree*.
 - 1.9) Модуль *WxMaskEdit* содержит реализацию вспомогательного класса *TWxLabeledMaskEdit*. Используется в модуле *WxEntry*.
 - 1.10) Модуль *WxMenuSettings* содержит реализацию формы для работы с настройками меню.
 - 1.11) Модуль *WxSynEdit* предоставляет доступ к возможностям библиотеки компонентов *SynEdit*.
- 2) Модули без GUI, реализующие другие возможности.
 - 2.1) Модуль *EpicTimer* содержит реализацию компонента таймера от *Tom Lisjac*. Описание компонента содержится на сайте <https://wiki.freepascal.org/EpikTimer>. Скачать компонент можно с сайта <https://github.com/graemeg/epiktimer>.
 - 2.2) Модуль *WxAppCfg* содержит функциональность по работе с расположением приложения на жёстком диске.
 - 2.3) Модуль *WxBits* содержит реализацию класса *TWxBits*, предназначенного для работы с отдельными битами блока памяти. Поддерживается до 2^{34} битов.

- 2.4) Модуль *WxBlinks* предназначен для работы с условно пустыми значениями переменных.
- 2.5) Модуль *WxCRC* содержит функциональность по вычислению контрольных сумм *CRC8*, *CRC32*, *CRC64*.
- 2.6) Модуль *WxDateUtils* содержит специальную функцию по переводу даты в строку.
- 2.7) Модуль *WxDbUtils* содержит набор вспомогательных функций для работы со стандартными классами *TField* и *TFieldDefs*.
- 2.8) Модуль *WxEncoding* содержит функциональность по работе с кодировками.
- 2.9) Модуль *WxKbd* содержит функциональность по переключению раскладок клавиатуры.
- 2.10) Модуль *WxLng* содержит языковые константы для возможности локализации *WXL*-приложений. Поддерживаются русский и английский языки.
- 2.11) Модуль *WxLogFile* содержит функциональность по работе с файлом журнала (лог-файлом).
- 2.12) Модуль *WxParamStr* является вспомогательным для модуля *WxAppCfg*. Содержит реализацию функций *HasOption* и *GetOptionValue* для проверки наличия опции и получения её значения.
- 2.13) Модуль *WxSort* содержит реализацию нескольких функций сортировки.
- 2.14) Модуль *WxStopWatch* содержит реализацию таймера *TWxStopWatch* для использования в других модулях *WXL*-библиотеки.
- 2.15) Модуль *WxStrSearch* содержит функциональность по поиску в строке и списке строк.
- 2.16) Модуль *WxStrUtils* содержит необходимые библиотеке *WXL* операции со строками.
- 2.17) Модуль *WxUnicode* позволяет работать с юникод-строками с учётом кодовых точек.
- 2.18) Модуль *WxVarUtils* содержит функциональность по работе с типом данных *variant*.

