Databases/ru

Эта страница является введением в темы Lazarus и **database**. В следующей таблице представлен обзор поддерживаемых баз данных.

Должны быть установлены только компоненты базы данных, для которых есть клиентские библиотеки (если базе данных нужны клиентские библиотеки), иначе Lazarus может не запуститься из-за отсутствующих файлов. Затем Lazarus необходимо переустановить, так как удаление компонента невозможно.

Databases portal

References:

- General info
- Libraries
- Field types
- Controls
- FAQ
- SQL how-to
- Working With TSQLQuery
- In-memory database applications

Tutorials/practical articles:

- Overview
- 0 - Database set-up
- 1 - Getting started
- 2 - Editing
- 3 - Queries
- 4 - Data modules
- SQLdb Programming Reference

Databases

Advantage - MySQL - MSSQL - Postgres - Interbase - Firebird - Oracle - ODBC - Paradox - SQLite - dBASE - MS Access - Zeos

Contents [hide]

Поддерживаемые базы данных

База данных	Имя пакета	Нужна клиентская библиотека?	Нужен сервер?	Поддерживаемые версии	Поддерживаемые платформы
Advantage	TAdsDataSet	Да	Нет	10.1 и больше	i386: Linux, Win32
DBase	DBFLaz	Нет	Нет	III+, IV, VII	Все
TurboPower FlashFiler	FlashFiler	Нет	Нет	-	Win 32, (win64?)
In memory	memds	Нет	Нет	-	Все
In memory	bufdataset	Нет	Нет	-	Все
Firebird	SQLdb	Да	Зависит от[1]	1 - 2.5	i386: Linux, Win32
(Visual) FoxPro	DBFLaz	Нет	Нет	2.0, 2.5, 3.0 (не полностью)	Все
Interbase	SQLdb	Да	Да	4 - 6	i386: Linux, Win32
Microsoft SQL Server	SQLdb	Да	Да	6-	FPC 2.6.2+. Linux, macOS, Win32, возможно *BSD, возможно Solaris[2]
MySQL	SQLdb	Да	Да	3.0 - 5.5	i386: Linux, Win32
ODBC	SQLdb	Да	Зависит от[3]	3.x	i386: Linux, Win32
Oracle	SQLdb	Да	Да	-	-
Paradox	TParadoxDataSet	Нет	Нет	до уровня таблицы 7 (и выше ??)	Все

База данных	Имя пакета	Нужна клиентская библиотека?	Нужен сервер?	Поддерживаемые версии	Поддерживаемые платформы
Paradox	TParadox	Да	Нет		Win32
PostgreSQL	SQLdb	Да	Да	6.6 - 8	i386: Linux, Win32
Sybase Adaptive Server Enterprise (ASE)	SQLdb	Да	Да	Любая	Linux, macOS, Win32, возможно *BSD, возможно Solaris[2)]
SQLite	SQLdb	Да	Нет	sqlite3	Все
SQLite	SQLite(3)Laz	Да	Нет	sqlite2,sqlite3	Все
Text files	sdf	Нет	Нет	-	Все

Примечание (1): Вы можете использовать встроенную версию Firebird для Windows и Linux (возможно, также для macOS) или подключиться к серверу Firebird, работающему на Windows/Unix/macOS/FreeBSD/других платформах, поддерживаемых Firebird.

Примечание (2): Эти коннекторы используют библиотеку FreeTDS в качестве драйвера. Документация FreeTDS указывает, что она должна быть построена как минимум на этих платформах. Версии Windows для x86 и x64 можно загрузить, например, отсюда отсюда (x32) и отсюда (x64)

Примечание (3): Этот номер версии относится к стандарту ODBC, а не к номеру версии драйвера или диспетчера драйверов. Для большинства СУБД существуют драйверы ODBC 3.x.

Привязка к клиентским библиотекам баз данных

Если вы хотите использовать одну из баз данных, которая требует клиентских библиотек, эти библиотеки должны быть установлены. Не только на компьютере, на котором вы программируете, но и на компьютерах, на которых должно работать приложение. Обратите внимание, что некоторые базы данных (в частности, MySQL) работают только в том случае, если привязки, скомпилированные в приложении, имеют ту же версию, что и у установленных библиотек. Вы можете узнать, как установить эти библиотеки (.so файлы в системах * nix и .dll в Windows) на веб-сайте разработчиков баз данных. Модули привязки можно найти в каталоге packages/base в fpc-sources. Они в основном состоят из клиентских API-вызовов, таких как mysql_connect_database, которые совершенно разные для каждой базы данных. Можно писать приложения баз данных, используя эти модули, но обычно это гораздо более трудоемко и критично к ошибкам, чем использование компонентов DB-модуля Lazarus.

Большинство из этих пакетов привязок жестко связаны с клиентскими библиотеками. Это означает, что, если приложение скомпилировано с одним из этих модулей, все приложение не может быть связано, если клиентские библиотеки недоступны на рабочей станции. В свою очередь, это означает, что исполняемый файл вашей программы не будет сгенерирован, если на вашем компьютере не установлен, например, клиент MySQL, и вы используете в своей программе модуль mysql4.pp. Если вам удастся скомпилировать программу на компьютере, на котором установлены клиентские библиотеки MySQL, она все равно не запустится ни на одном другом компьютере без соответствующих клиентских библиотек MySQL. Другими словами: для этих баз данных вам нужно установить клиентские библиотеки на компьютере разработчика и установить эти клиентские библиотеки вместе с вашим приложением.

Чтобы избежать таких проблем, некоторые пакеты также могут динамически подключаться к библиотекам. Перед выполнением каких-либо вызовов этих библиотек устройство должно быть 'инициализировано'. Эта инициализация завершается ошибкой, если на компьютере не установлен клиент базы данных. Если программа готова использовать клиентскую библиотеку, модуль должен быть 'освобожден'.

Наборы данных (Dataset)

Базы данных в Lazarus (или FreePascal) основывают свою работу на базовом классе TDataset. Этот класс представляет таблицу или результат запроса в Вашем приложении. Как и многие другие базовые классы, Вы не

можете использовать TDataSet в своём приложении непосредственно, а используете только его потомков, которых довольно много. Они обеспечивают доступ к различным видам баз данных, таким как локальный dbase или текстовые файлы, а так же серверами баз данных, таких как PostgreSQL, Firebird, MySQl и т.п. Некоторые потомки TDataSet связываются непосредственно с таблицами базы данных, в то время как другие используют дополнительные компоненты или библиотеки.

Обратитесь к странице базы данных для получения более полной информации об этом.

Потомки Dataset, будучи не визуальными компонентами, обычно являются частью свободной библиотеки компонентов (FCL), а не библиотеки компонентов Lazarus (LCL).

Использование наборов данных в коде программы

О программном доступе объясняется более подробно в "Использование компонентов Dataset и Field", вот краткий обзор:

- Используйте потомок TDataset чтобы открыть таблицу или запрос, отфильтруйте те строки, которые хотите видеть и двигайтесь от строки к строке.
- Используйте потомок TField для:
  - Доступа к общей информации о поле.
  - Доступа к данным текущей строки. (используя свойство AsXXXX, например AsString, AsInteger и так далее.)
- Используйте для обращения к полям потомка TDatasetтак же:
  - Свойство Fields, к примеру Fields[0] для самого первого поля,
  - Метод FieldByName, как пример FieldByName('AGE') вернёт поле, которое в базе данных называется 'AGE'.

См. Тип поля базы данных для просмотра списка типов полей.

Использование визуальных (db-ориентированных) компонентов

Чтобы использовать базы данных в простом, приложении Lazarus в "RAD"-стиле, обычно настраивается потомок TDataSet во время разработки и добавляются специализированные компоненты. Например:

- Добавьте потомка dataset для выбранной Вами базы данных вместе с другими необходимыми компонентами на Вашу форму и откройте его (Установите свойство 'Active' в True).
- Добавьте на форму компонент TDatasource (с вкладки "Data Access") и свяжите его с датасетом (установив свойство DataSet).
- Добавьте компоненты отображения данных с вкладки "Data Controls" и свяжите их с компонентом DataSource (**не** dataset).
- Большинство компонентов связываются только с одним полем, поэтому Вам надо будет установить свойство Field для каждого такого компонента.

Смотрите Элементы управления данными ниже.

Рабочие состояния (Статусы) Dataset

Наборы данных могут быть в нескольких состояниях (Статусах), которые отображаются в свойстве State. Вот какие состояния могут быть (смотрите TDataSetState в исходниках):

Описание	Функция
dsInactive	Набор данных закрыт, доступ к данным невозможен
dsBrowse	В этом состоянии можно просматривать набор данных, искать какие-либо значения
dsEdit	В этом состоянии можно редактировать данные в текущей строке. Изменённые значения не будут записаны, пока не применён метод Post.
dsInsert	В этом состоянии только что вставлена новая строка данных и туда можно заносить новые значения. Строка не будет записана, пока не применён метод Post.

Другие состояния обычно являются кратковременными и обрабатываются автоматически. Они используются внутри компонента и более сложны в коде обработки. Если Ваше приложение только просматривает данные и Вы

открываете набор данных во времени разработки, то Вы можете просто проигнорировать статус, поскольку он главным образом будет dsBrowse. Однако большинство приложений будут тем или иным образом всё-таки изменять данные. Если Вы используете DB-ориентированные компоненты, то они обрабатывают эти состояния автоматически. Если Вы измените текст в компоненте (к примеру) TDBEdit, то он пометит набор данных в состояние dsEdit, если только Вы уже не поставили вручную статус в dsEdit или dsInsert. Если Вы будете переходить от одной записи к другой, то та запись, которая была в состоянии dsEdit или dsInsert, автоматически выполнит метод Post, при переходе на новую строку и DataSet вернётся в состояние dsBrowse. Однако если Вы обращаетесь к набору данных в коде, то его состояние Вы должны выставлять вручную. Компонент TDBNavigator позволяет изменять статус самому пользователю.

UpdateStatus набора данных

UpdateStatus определяет текущее состояние буфера записи, если обновления еще не были применены к базе данных.

Пример, как определить, будет ли ApplyUpdates вставлять, обновлять или удалять данные:

```
procedure QueryAfterPost(DataSet: TDataSet);
begin
  case DataSet.UpdateStatus of
    usUnmodified : ShowMessage('Unmodified');
    usModified   : ShowMessage('Modified');
    usInserted   : ShowMessage('Inserted');
    usDeleted    : ShowMessage('Deleted');
  end;
end;
```

Объяснение значений

- usUnmodified: Запись не изменялась
- usModified: Запись существует в базе данных, но локально изменена
- usInserted: Запись еще не существует в базе данных, но вставлена локально
- usDeleted: Запись еще существует в базе данных, но локально удалена

Принятие и отмена изменения

Если Вы редактируете или вставляете новую запись, то значения этой записи сначала помещаются в буфер. После этого:

- Вызов метода `Dataset.Cancel` удаляет новую запись (при вставке) или или возвращает старые значения записи (при редактировании).
- Вызов метода `Dataset.Post` сохраняет значения (при редактировании) или новую запись (при вставке).

В некоторых потомках Dataset значения будут немедленно написаны базе данных, в то время как в других они будут сохранены во временной таблице обновлений, пока не будет вызван некий окончательный метод, чтобы сохранить все изменения в базе данных. Однако иногда, даже когда значения записаны в базу данных, Вам, вероятно, придется дополнительно вызвать метод *Commit*, чтобы заставить базу данных принять эти изменения. Всё это, в значительной степени, зависит от конкретного потомка DataSet'а, поэтому за подробностями обращайтесь к описанию этого самого конкретного потомка.

Вставка новой записи

Чтобы вставить новую запись в потомок TDataSet, следует использовать метод Insert . После этого можно установить значения полей и, наконец, вызвать Post для фиксации новой записи, как показано в примере ниже.

В примере также показано, как вставить BLOB-данные из файла - вы также можете использовать LoadFromStream для загрузки данных из потока.

```
  MyDataset.Insert;
  MyDataset.Fields[0].AsInteger := 4; //целочисленное поле
  MyDataset.Fields[1].AsString := 'First Name'; //строковое поле
```

```
  TBlobField(MyDataset.Fields[2]).LoadFromFile('SomeBlobfile.bin'); //blob-поле
  MyDataset.Post;
```

Как быстро перейти к одной конкретной записи в таблице

После селекта всех записей таблицы

Если вы используете `SELECT * FROM`, чтобы выбрать все записи таблицы, а затем хотите быстро переключаться между ними, вам придется построить индекс и выполнить поиск по нему. Более эффективно выбрать только ту запись, которую вы хотите.

Селект только нужной записи

Одним из быстрых решений для перехода к определенной записи является выбор только ее, например:

```
var
  MyDataset: TSQLQuery;
begin
  //...
  MyDataset.FieldDefs.Add('SessionId', ftLargeint);
  MyDataset.FieldDefs.Add('GameEvent', ftLargeint);
  MyDataset.FieldDefs.Add('TableId', ftInteger);
  MyDataset.FieldDefs.Add('LoggedIn', ftBoolean);
  MyDataset.FieldDefs.Add('PlayerId', ftInteger);
  MyDataset.Active := False;
  { Непараметризированный формат; могут возникнуть проблемы с текстом, содержащим
одинарные кавычки "'" и даты
  SQLText := Format('select * from "GameSession" WHERE "SessionId"=%d', [ASessionId]);}

  // Решение: параметризированный запрос:
  //На самом деле, если это делается в цикле, вам нужно задать SQL.Text один раз,
  // и дальше менять только значение параметра
  MyDataset.SQL.Text := 'select * from "GameSession" WHERE "SessionID"=:SessionID';
  MyDataSet.ParamByName('SessionID').AsLargeInt := ASessionID;
  try
    MyDataset.Active := True;
  except
    //...
  end;
```

Затем вы можете прочитать информацию, используя что-то вроде этого:

```
  lPlayerId := MyDataset.Fields[4].AsInteger;
```

Фильтрация

Вы можете отфильтровать ваш набор данных, чтобы ограничить записи нужным подмножеством (например, все фамилии, начиная с Иванова).

- Используя .Filter:
  - TDbf, TBufDataset и потомки (включая TSQLQuery)- используйте анализатор фильтрации TDBF; смотрите Expressions для более подробного объяснения.
  - TMemDataset - не поддерживает .Filter
- Используя процедуры callback/события OnFilter, где вы можете запрограммировать свою собственную функцию фильтра

Locate/lookup

Хотя чаще используются в не-SQL наборах данных (например, TParadoxDataSet, TDbf), вы можете перемещаться между записями, используя locate/lookup.

Использование TSQLQuery

Для более подробной информации о TSQLQuery см. Working With TSQLQuery

Экспорт

FPC/Lazarus содержит функциональность, позволяющую экспортировать наборы данных в различные форматы; см. например

- fpXMLXSDExport
- fpdbfexport
- другие компоненты на вкладке Data Export (Экспорт данных)

Конечно, вы также можете сделать это вручную (см., Например, Преобразование базы данных в электронную таблицу для экспорта в формат Excel с использованием fpspreadsheet)

Элементы управления данными

Чтобы использовать любой из этих элементов управления, добавьте элемент управления на форму и задайте минимально хотя бы свойство источника данных. Другие ключевые свойства будут заполнены автоматически.

Элемент управления Datasource

Этот элемент управления отслеживает, какие записи связанных элементов управления в данный момент включены. Элемент управления Datasource должен быть связан с набором данных (например, TSQLQuery).

Элемент управления одним полем

Все эти элементы управления прикрепляются к одному полю. Как и с datasource, задайте имя поля. Элементы управления включают в себя:

- Элемент управления DBText - Отображает текстовое поле (только чтение, без обрамления)
- Элемент управления DBEdit - Отображает / редактирует текстовое поле как edit box
- Элемент управления DBMemo - Отображает / редактирует текстовое поле как многострочный edit box
- Элемент управления DBImage - Показывает изображение, хранящееся в базе данных, как BLOB. Примечание: по умолчанию Lazarus сохраняет заголовок с типом изображения перед данными изображения в поле BLOB базы данных. Этим он отличается от Delphi. Однако вы можете сделать TDBImage Delphi-совместимым: см. TDBImage
- Элемент управления DBListBox и элемент управления DBComboBox позволяют пользователю вставлять значения в поле базы данных из списка значений в свойстве Items элементов управления
- Элемент управления DBLookupListBox и элемент управления DBLookupComboBox, см. также TDBLookupComboBox - позволяют пользователю вставлять значения в поле базы данных, отображая содержимое поля поиска в другой таблице. Хотя эти элементы управления хранят свои результаты в одном поле, вам нужно другое поле для поиска значений. **Примечание**: по крайней мере для DBLookupComboBox, есть ошибка с FPC 2.6.0, которая требует, чтобы поле списка (listfield) также присутствовало в datasource; вы можете обойти его, объявив вычисляемое поле с тем же именем, что и поле списка в наборе данных datasource, которое ничего не делает.
- Элемент управления DBCheckBox - Отображает / редактирует логическое поле путем установки / снятия флажка (крыжика)
- Элемент управления DBRadioGroup - Отображает элементы как в обычной группе радиокнопок, считывая/ устанавливая значение поля из списка совпадающих значений
- Элемент управления DBCalendar - Отображает / редактирует поле даты с помощью панели календаря
- Элемент управления DBGroupBox - версия TGroupBox с поддержкой данных, позволяющая группировать несколько объектов с данными на форме

Элемент управления DBGrid

Этот элемент управления может отображать несколько полей в макете строки/столбца - фактически по умолчанию он показывает их все. Однако вы можете поместить записи в columns collection (список столбцов), чтобы ограничить его определенными полями и установить ширину и заголовки каждого столбца.

Помимо упомянутой документации, некоторые подробности можно найти здесь: TCustomDBGrid

Элемент управления Navigator

Этот элемент управления дает пользователю непосредственный контроль над набором данных. Это позволяет пользователю:

- переходить к следующей или предыдущей записи, или к началу или концу записей
- добавлять новую запись (эквивалентно вызову метода dataset.insert)
- переводить набор данных в режим редактирования
- удалять запись
- подтверждать или отменять текущие изменения
- обновлять данные (полезно в многопользовательских приложениях базы данных)

Ключевые свойства:

- VisibleButtons: позволяет контролировать, что может делать пользователь. Например, если удаление не разрешено, скрыть кнопку удаления. Если у вас есть DBGrid, присоединенный к тому же набору данных, вы можете решить, что вам не нужны следующие и предыдущие кнопки.
- Width: если вы показываете не все кнопки, вы можете установить ширину (высота * кол-во_видимых_кнопок)

Выполнение тестов базы данных FPC

Бесплатные компоненты баз данных Pascal включают, основанную на fpcunit, тестовую среду dbtestframework, которую можно использовать для проверки функциональности. Смотрите каталог *source\packages\fcl-db\tests* в дереве исходного кода FPC. Туда включен тестовый фреймворк, который может быть запущен на различных компонентах базы данных, а также некоторые другие тесты (например, тест экспорта базы данных).

Чтобы запустить тестовую среду для определенной базы данных:

1. Сохраните файл *source\packages\fcl-db\tests\database.ini.txt* как *source\packages\fcl-db\tests\database.ini*

2. Измените файл *source\packages\fcl-db\tests\database.ini*, чтобы выбрать какой тип базы данных вы будете использовать.

Пример для Interbase/Firebird:

```
[Database]
type=interbase
```

3. В том же файле настройте параметры для вашей базы данных. Например. если вы ранее выбрали interbase:

```
[interbase]
connector=sql
connectorparams=interbase
; Имя базы данных/путь (примечание: база данных уже должна существовать)
; Вы можете использовать псевдонимы (см. aliases.conf к вашей документации Firebird)
name=testdb
user=sysdba
password=masterkey
; Ваше имя хоста может очень отличаться:
; Оставьте пустым, если вы хотите использовать базу данных embedded Firebird
hostname=192.168.0.42
```

(прим.перев.: если сервер запущен на вашей машине, то обычно используют loopback ("сетевую петлю"), т.е. hostname указывается как *localhost* или 127.0.0.1)

---

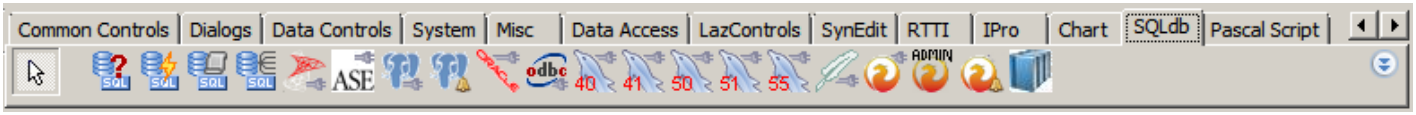
4. Скомпилируйте и запустите *source\packages\fcl-db\tests\dbtestframework.pas* (вы также можете использовать Lazarus для компиляции и запуска версии с графическим интерфейсом, dbtestframework_gui). Если вы используете встроенную базу данных в Windows (например, embedded Firebird или sqlite), сначала скопируйте необходимые DLL-

файлы в каталог. Вывод будет в формате XML (или будет отображаться на вашем экране, если вы используете dbtestframework_gui).

Пожалуйста, см. *source\packages\fcl-db\tests\README.txt* для получения более подробной информации.

Пакеты базы данных, содержащиеся в Lazarus

sqldblaz.lpk



Этот пакет обеспечивает доступ к различным базам данных. К ним относятся:

- Interbase/Firebird
- Microsoft SQL Server (за исключением Lazarus/FPC x64 для Windows)
- MySQL
- Oracle (за исключением Lazarus/FPC x64 для Windows)
- PostgreSQL (за исключением Lazarus/FPC x64 для Windows)
- SQLite (с поддержкой расширения Spatialite)
- Sybase ASE (Adaptive Server Enterprise - не следует путать с Sybase ASA) (за исключением Lazarus/FPC x64 для Windows)
- любая база данных с драйвером ODBC.

Компоненты (TSQLQuery, TSQLTransaction, TIBConnection, TODBCConnection, TOracleConnection, TMSSQLConnection, TMySQL40Connection, TMySQL41Connection, TMySQL50Connection, TPQConnection, TSybaseConnection) находятся на вкладке 'SQLdb' в палитре компонентов.

- Пакет SQLdb

dbflaz.lpk

Этот пакет обеспечивает доступ к базам данных dBase и FoxPro. Вы можете получить больше информации в Руководстве по Lazarus Tdbf. Компонент TDbf находится на вкладке Data Access в палитре компонентов.

sqlitelaz.lpk

Этот пакет предоставляет доступ к базам данных SQLite. Вы можете получить больше информации в разделе Обзор баз данных Lazarus.

sdflaz.lpk

Компонент TSdfDataSet может быть найден на вкладке Data Access в палитре компонентов.

lazreport.lpk

Домашняя страница генератора отчетов: здесь . Больше информации (и др. дополнительные ссылки) можно найти здесь. LazReport зависит от пакета Printer4Lazarus. Начиная с ревизии 11950 LazReport был включен в репозиторий Lazarus SVN.

lazdbexport.lpk

См. lazdbexport.

Внешние пакеты / библиотеки

Zeos DataBase Objects

Эти компоненты обеспечивают доступ к различным базам данных. Вы можете найти больше информации здесь. Эта вики также содержит руководство по Zeos.

Pascal Data Objects

Теперь есть альтернатива.

Поддерживаются:

- MySQL 4.1 и 5.0
- sqlite-2 и sqlite-3
- pgsql-8.1
- interbase-5, interbase-6, firebird-1.0, firebird-1.5, firebird-1.5E, firebird-2.0, firebird-2.0E
- mssql (Microsoft library) и sybase (FreeTDS library)
- oracle

как подготовленные операторы, привязка и хранимые процедуры поддерживаются API базы данных, который называется Pascal Data Objects, который вдохновлен PHP Data Objects. Весь код и документация, необходимые для использования этого нового API, доступны на Sourceforge: http://pdo.sourceforge.net/

TPSQL

Эти компоненты обеспечивают доступ к базам данных PostgreSQL посредством TCP/IP. Вы можете найти больше информации на этой странице.

FIBL

Эти компоненты обеспечивают доступ к базам данных Interbase и Firebird. Домашняя страница http://sourceforge.net/projects/fibl .

IBX

IBX For Lazarus - это компоненты для доступа к базам данных Firebird: см. IBX

FBLib Firebird Library

FBLib является библиотекой с открытым исходным кодом, не-Data Aware, для прямого доступа к реляционной базе данных Firebird от Borland Delphi/Kylix, Free Pascal и Lazarus.

Текущие функции включают в себя:

- Прямой доступ к Firebird 1.0.x, 1.5.x и 2.x Classic или SuperServer
- Мультиплатформа [Win32, Gnu/Linux, FreeBSD)
- Автоматический выбор клиентской библиотеки 'fbclient' или 'gds32'
- Запрос с параметрами
- Поддержка диалекта SQL 1/3
- Лицензионное соглашение LGPL
- Извлечение метаданных
- Парсер простого сценария
- В окончательный EXE-файл добавляется только 100-150 КБ
- Поддержка BLOB-полей
- Экспорт данных в HTML-скрипт SQL
- Диспетчер служб (резервное копирование, восстановление, исправление ...)
- Оповещение событий

Вы можете загрузить документацию с github .

Unified Interbase

UIB обеспечивает доступ к базам данных Interbase, Firebird и YAFFIL. Домашняя страница на sourceforge.net . Репозиторий SVN доступен по адресу https://github.com/hgourvest/uib

TechInsite Object Persistence Framework (tiOPF)

Более подробную информацию о tiOPF можно найти на этойстранице.

Advantage TDataSet Descedant

Advantage TDataSet Descedant предоставляет средства для подключения (и открытия таблиц) к серверу базы данных Advantage. Advantage - это гибкая, не требующая администрирования встроенная база данных, которая предоставляет клиент-сервер, а также одноранговый доступ к форматам файлов Clipper, FoxPro и Visual FoxPro 9 DBF, и также собственный формат файлов, обеспечивающий путь миграции, позволяющий использовать новые функции.

Основные характеристики:

- Бесплатный доступ к одноранговой базе данных с возможностью перехода на клиент/сервер
- Многоплатформенность (клиенты поддерживаются в Windows и Linux, сервер поддерживается в Windows, Linux и NetWare)
- Поддерживает как навигационный, так и реляционный SQL доступ к базе данных
- Полнотекстовая поисковая система
- Шифрование таблиц, индексов, memo-полей и передаваемых данных
- Совместимость с нативными компонентами TDataset
- Резервное копирование в онлайн-хранилище
- Сервер поддерживает репликацию

Для получения дополнительной информации см. домашнюю страницу Advantage Database Server .

ZMSQL, sql-расширяемая база данных в памяти

Для получения дополнительной информации см. ZMSQL wiki-страницу ZMSQL

ZMSQL - это база данных с открытым исходным кодом, наследуемая от TBufDataset, sql-расширяемая база данных в памяти для Free Pascal (FPC), работающая со значениями, разделяемых точкой с запятой, плоских текстовых таблиц. Полностью написанная на Pascal она не зависит от внешних библиотек. Использует движок JanSQL для реализации SQL.

Она предлагает:

- Загрузку из и сохранение данных в виде плоских текстовых таблиц
- Использование SQL-запросов для получения данных
- Копирование данных и схем из других наборов данных
- Возможность предопределения fielddefs или создание их на лету
- Механизм фильтрации Master/detail
- Ссылочную целостность
- Параметризованные запросы

Загружаемый архив содержит исходный код, некоторые демонстрационные приложения, иллюстрирующие функции компонента, а также файл readme.

См.также

- Database Portal
- Lazarus DB Faq

Categories: Russian | FPC/ru | Lazarus/ru | Databases/ru | Testing/ru