Содержание

Введение

Глава 1. MySQL

1.1 Исследовательская часть

1.2 SQL

1.3 Основные характеристики

1.4 Требования к аппаратному обеспечению

Глава 2. Конструкторская часть

2.1 Создание таблицы

2.2 Создание триггеров

2.3 Создание генераторов

2.4 Создание Хранимых процедур

2.5 Разработка приложения

2.6 Блок схема

Глава 3. Руководство пользователю

Заключение

Список литературы

#### Введение

SQL является, прежде всего, информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. SQL нельзя назвать языком программирования

Изначально, SQL был основным способом работы пользователя с базой данных и позволял выполнять следующий набор операций:

* создание в базе данных новой таблицы;
* добавление в таблицу новых записей;
* изменение записей;
* удаление записей;
* выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);

а, также, изменение структур таблиц. Со временем, SQL усложнился — обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (например, индексы, представления, триггеры и хранимые процедуры) — и стал приобретать черты, свойственные языкам программирования.

При всех своих изменениях, SQL остаётся единственным механизмом связи между прикладным программным обеспечением и базой данных. В то же время, современные СУБД, а, также, информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

Каждое предложение SQL — это запрос или обращение к базе данных, которое приводит к изменению в базе данных. В соответствии с тем, какие изменения происходят в базе данных, различают следующие типы запросов:

* запросы на создание или изменение в базе данных новых или существующих объектов (при этом в запросе описывается тип и структура создаваемого или изменяемого объекта);
* запросы на получение данных;
* запросы на добавление новых данных (записей)
* запросы на удаление данных;
* обращения к СУБД.

Основным объектом хранения реляционной базы данных является таблица, поэтому все SQL-запросы — это операции над таблицами. В соответствии с этим, запросы делятся на

* запросы, оперирующие самими таблицами (создание и изменение таблиц);
* запросы, оперирующие с отдельными записями (или строками таблиц) или наборами записей.

Каждая таблица описывается в виде перечисления своих полей (столбцов таблицы) с указанием

* типа хранимых в каждом поле значений;
* связей между таблицами (задание первичных и вторичных ключей);
* информации, необходимой для построения индексов.

Запросы первого типа, в свою очередь, делятся на запросы, предназначенные для создания в базе данных новых таблиц, и на запросы, предназначенные для изменения уже существующих таблиц. Запросы второго типа оперируют со строками, и их можно разделить на запросы следующего вида:

* вставка новой строки;
* изменение значений полей строки или набора строк;
* удаление строки или набора строк.

Самый главный вид запроса — это запрос, возвращающий (пользователю) некоторый набор строк, с которым можно осуществить одну из трёх операций:

* просмотреть полученный набор;
* изменить все записи набора;
* удалить все записи набора.

Таким образом, использование SQL сводится, по сути, к формированию всевозможных выборок строк и совершению операций над всеми записями, входящими в набор.

В начале 1970-х годов в одной из исследовательских лабораторий компании IBM была разработана экспериментальная реляционная СУБД IBM System R, для которой затем был создан специальный язык SEQUEL, позволявший относительно просто управлять данными в этой СУБД. Аббревиатура SEQUEL расшифровывалась как Structured English QUEry Language — «структурированный английский язык запросов». Позже по юридическим соображениям язык SEQUEL был переименован в SQL. Когда в 1986 году первый стандарт языка SQL был принят ANSI (American National Standards Institute), официальным произношением стало [,es kju:' el] — эс-кью-эл. Несмотря на это, даже англоязычные специалисты зачастую продолжают читать SQL как сиквел (по-русски также часто говорят «эс-ку-эль» или используют жаргонизм «скуль»).

Целью разработки было создание простого непроцедурного языка, которым мог воспользоваться любой пользователь, даже не имеющий навыков программирования. Собственно разработкой языка запросов занимались Дональд Чэмбэрлин (Donald D. Chamberlin) и Рэй Бойс (Ray Boyce). Пэт Селинджер (Pat Selinger) занималась разработкой стоимостного оптимизатора (cost-based optimizer), Рэймунд Лори (Raymond Lorie) занимался компилятором запросов.

Стоит отметить, что SEQUEL был не единственным языком подобного назначения. В Калифорнийском Университете Беркли была разработана некоммерческая СУБД Ingres (являвшаяся, между прочим, дальним прародителем популярной сейчас некоммерческой СУБД PostgreSQL), которая являлась реляционной СУБД, но использовала свой собственный язык QUEL, который, однако, не выдержал конкуренции по количеству поддерживающих его СУБД с языком SQL.

Первыми СУБД, поддерживающими новый язык, стали в 1979 году Oracle V2 для машин VAX от компании Relational Software Inc. (впоследствии ставшей компанией Oracle) и System/38 от IBM, основанная на System/R.

Глава 1. MySQL

1.1 Исследовательская часть

база данные borland delphi

База данных представляет собой структурированную совокупность данных. Эти данные могут быть любыми - от простого списка предстоящих покупок до перечня экспонатов картинной галереи или огромного количества информации в корпоративной сети. Для записи, выборки и обработки данных, хранящихся в компьютерной базе данных, необходима система управления базой данных, каковой и является ПО MySQL. Поскольку компьютеры замечательно справляются с обработкой больших объемов данных, управление базами данных играет центральную роль в вычислениях. Реализовано такое управление может быть по-разному - как в виде отдельных утилит, так и в виде кода, входящего в состав других приложений.

MySQL - это система управления реляционными базами данных. В реляционной базе данных данные хранятся не все скопом, а в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости. Таблицы связываются между собой при помощи отношений, благодаря чему обеспечивается возможность объединять при выполнении запроса данные из нескольких таблиц. **SQL** как часть системы MySQL можно охарактеризовать как язык структурированных запросов плюс наиболее распространенный стандартный язык, используемый для доступа к базам данных.

MySQL - это ПО с открытым кодом. Применять его и модифицировать может любой желающий. Такое ПО можно получать по Internet и использовать бесплатно. При этом каждый пользователь может изучить исходный код и изменить его в соответствии со своими потребностями.

Почему веб-программисты отдают предпочтение СУБД MySQL? MySQL является очень быстрым, надежным и легким в использовании. Если вам требуются именно эти качества, попробуйте поработать с данным сервером. MySQL обладает также рядом удобных возможностей, разработанных в тесном контакте с пользователями. Первоначально сервер MySQL разрабатывался для управления большими базами данных с целью обеспечить более высокую скорость работы по сравнению с существующими на тот момент аналогами. И вот уже в течение нескольких лет данный сервер успешно используется в условиях промышленной эксплуатации с высокими требованиями. Несмотря на то что MySQL постоянно совершенствуется, он уже сегодня обеспечивает широкий спектр полезных функций. Благодаря своей доступности, скорости и безопасности MySQL очень хорошо подходит для доступа к базам данных по Internet.

Технические возможности СУБД MySQL

MySQL является системой клиент-сервер, которая содержит многопоточный SQL-сервер, обеспечивающий поддержку различных вычислительных машин баз данных, а также несколько различных клиентских программ и библиотек, средства администрирования и широкий спектр программных интерфейсов (API). Мы также поставляем сервер MySQL в виде многопоточной библиотеки, которую можно подключить к пользовательскому приложению и получить компактный, более быстрый и легкий в управлении продукт. Доступно также большое количество программного обеспечения для MySQL, в большей части - бесплатного.

MySQL правильно произносится как ''Май Эс Кью Эль''.

## SQL

SQL состоит из двух частей: серверной и клиентской.

Сервер MySQL постоянно работает на компьютере. Клиентские программы (например, скрипты PHP) посылают серверу MySQL SQL-запросы через механизм сокетов (то есть при помощи сетевых средств), сервер их обрабатывает и запоминает результат. То есть скрипт (клиент) указывает, какую информацио он хочет получить от сервера баз данных. Затем сервер баз данных посылает ответ (результат) клиенту (скрипту).

Почему всегда передается не весь результат? Очень просто: дело в том, что размер результирующего набора данных может быть слишком большим, и на его передачу по сети уйдет чересчур много времени. Да и редко когда бывает нужно получать сразу весь вывод запроса (то есть все записи, удовлетворяющие выражению запроса). Например, нам может потребоваться лишь подсчитать, сколько записей удовлетворяет тому или иному условию, или же выбрать из данных только первые 10 записей. Механизм использования сокетов подразумевает технологию клиент-сервер, а это означает, что в системе должна быть запущена специальная программа — MySQL-сервер, которая принимает и обрабатывает запросы от программ. Так как вся работа происходит в действительности на одной машине, накладные расходы по работе с сетевыми средствами незначительны (установка и поддержание соединения с MySQL-сервером обходится довольно дешево).

Cтруктура MySQL трехуровневая: базы данных — таблицы — записи. Базы данных и таблицы MySQL физически представляются файлами с расширениями frm, MYD, MYI. Логически - таблица представляет собой совокупность записей. А записи - это совокупность полей разного типа. Имя базы данных MySQL уникально в пределах системы, а таблицы - в пределах базы данных, поля - в пределах таблицы. Один сервер MySQL может поддерживать сразу несколько баз данных, доступ к которым может разграничиваться логином и паролем. Зная эти логин и пароль, можно работать с конкретной базой данных. Например, можно создать или удалить в ней таблицу, добавить записи и т. д. Обычно имя-идентификатор и пароль назначаются хостинг провайдерами, которые и обеспечивают поддержку MySQL для своих пользователей.

## 1.3 Основные характеристики

* MySQL - это система управления реляционными базами данных.

В реляционной базе данные хранятся в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости. Таблицы связываются между собой при помощи отношений, благодаря чему обеспечивается возможность объединять при выполнении запроса данные из нескольких таблиц. SQL как часть системы MySQL можно охарактеризовать как язык структурированных запросов плюс наиболее распространенный стандартный язык, используемый для доступа к базам данных.

* Программное обеспечение MySQL - это ПО с открытым кодом.

ПО с открытым кодом означает, что применять и модифицировать его может любой желающий. Такое ПО можно получать по Internet и использовать бесплатно. При этом каждый пользователь может изучить исходный код и изменить его в соответствии со своими потребностями.

* Технические возможности СУБД MySQL

ПО MySQL является системой клиент-сервер, которая содержит многопоточный SQL-сервер, обеспечивающий поддержку различных вычислительных машин баз данных, а также несколько различных клиентских программ и библиотек, средства администрирования и широкий спектр программных интерфейсов (API).

* Безопасность

Система безопасности основана на привилегиях и паролях с возможностью верификации с удаленного компьютера, за счет чего обеспечивается гибкость и безопасность. Пароли при передаче по сети при соединении с сервером шифруются. Клиенты могут соединяться с MySQL, используя сокеты TCP/IP, сокеты Unix или именованные каналы (named pipes, под NT)

* Вместимость данных

Начиная с MySQL версии 3.23, где используется новый тип таблиц, максимальный размер таблицы доведен до 8 миллионов терабайт (263 bytes). Однако следует заметить, что операционные системы имеют свои собственные ограничения по размерам файлов. Ниже приведено несколько примеров:

- 32-разрядная Linux-Intel – размер таблицы 4 Гб.

- Solaris 2.7 Intel - 4 Гб

- Solaris 2.7 UltraSPARC - 512 Гб

- WindowsXP - 4 Гб

Как можно видеть, размер таблицы в базе данных MySQL обычно лимитируется операционной системой. По умолчанию MySQL-таблицы имеют максимальный размер около 4 Гб. Для любой таблицы можно проверить/определить ее максимальный размер с помощью команд SHOW TABLE STATUS или myisamchk -dv table\_name. Если большая таблица предназначена только для чтения, можно воспользоваться myisampack, чтобы слить несколько таблиц в одну и сжать ее. Обычно myisampack ужимает таблицу по крайней мере на 50%, поэтому в результате можно получить очень большие таблицы.

## 1.4 Требования к аппаратному обеспечению

Система предъявляет следующие требования к аппаратному и программному обеспечению сервера:

Минимальные требования к серверу: 1 компьютер с процессором Pentium II 233 МГц и выше, 128 Мб оперативной памяти.

Рекомендуемые требования к серверу: 2 компьютера с процессором Pentium II 233 МГц и выше, 256Мб оперативной памяти. В этом случае предполагается, что один компьютер будет выполнять функции SQL сервера, а другой - сервера приложений CimWebCenter.

Для работы системы необходимо следующее предустановленное программное обеспечение: Apache Web Server , PHP 4.3.x, MySql , на платформе FreeBSD или Linux.

Существует также возможность инсталляции системы на платформе Windows. При этом требования к предустановленному программному обеспечению остаются прежними.

Для работы с клиентской частью системы требуется компьютер, подключенный по протоколу TCP/IP к сети, в которой находится сервер с установленной CimWebCenter.

**Минимальные требования к компьютеру клиента:** Pentium 200 МГц и выше, 32 Мб оперативной памяти. Работа с системой осуществляется посредством браузера Internet Explorer версии 6.0.

**Для работы в режиме WYSIWIG редактора,** также требуется наличие на клиентском компьютере установленного пакета **MsOffice**. При первом запуске системы может потребоваться **дистрибутив пакета MsOffice**.

Глава 2. Конструкторская часть

* Создание базы данных.
* Для создания базы данных использовался Ibexpert:
* Утилита администрации InterBase (поддерживет InterBase MySQL Yaffil) Работает одновременно с несколькими базами данных. Содержит редакторы для всех объектов БД с синтаксической подсветкой, SQL редактор с историей запросов и возможностью фонового выполнения запросов, отладчик хранимых процедур и триггеров, поиск в метаданных, полное и частичное извлечение данных и метаданных, анализатор зависимостей объектов баз данных, отчеты по метаданным, менеджеры пользователей и пользовательских привилегий, экспорт данных в различные форматы.

Данный триггер следит за изменением, если пользователь изменил любое поле из таблицы goroda, триггер автоматически записывает в таблицу LOG имя пользователя и дату.

#### 2.1 Создание таблицы

Таблица – основной объект для хранения информации в реляционной базе данных. Она состоит из содержащих данные строк и столбцов, занимает в базе данных физическое пространство и может быть постоянной или временной. Поле, также называемое в реляционной базе данных столбцом, является частью таблицы, за которой закреплен определенный тип данных. Каждая таблица базы данных должна содержать хотя бы один столбец. Строка данных – это запись в таблице базы данных, она включает поля, содержащие данные из одной записи таблицы.

Главное в команде создания таблицы – определение имени таблицы и описание набора имен полей, которые указываются в соответствующем порядке. Кроме того, этой командой оговариваются типы данных и размеры полей таблицы.

Ключевое слово NULL используется для указания того, что в данном столбце могут содержаться значения NULL. Значение NULL отличается от пробела или нуля – к нему прибегают, когда необходимо указать, что данные недоступны, опущены или недопустимы. Если указано ключевое слово NOT NULL, то будут отклонены любые попытки поместить значение NULL в данный столбец. Если указан параметр NULL, помещение значений NULL в столбец разрешено. По умолчанию стандарт SQL предполагает наличие ключевого слова NULL.

CREATE GENERATOR GEN\_GORODA\_ID;

CREATE TABLE GORODA (

ID\_GOR INTEGER NOT NULL,

NAZVANIE VARCHAR(30) NOT NULL,

DOLGOTA INTEGER,

SHIROTA INTEGER,

ID\_STR INTEGER NOT NULL);

с помощью этого кода создается таблица города, она состоит из следующих столбцов:

id\_gor – индентефикатор города, данное поле не может быть пустым.

nazvanie- это поле содержит название города, оно не может превышать 30 символов.

id\_str – индентификатор страны к которой принадлежит город.

dolgota- долгота на которой находится данный город.

shirota- широта на которой находится данный город.

CREATE GENERATOR GEN\_STRANA\_ID\_STR;

CREATE TABLE STRANA (

ID\_STR INTEGER NOT NULL,

NAZVANIE VARCHAR(30) NOT NULL,

STOLICA VARCHAR(30) NOT NULL,

NASELENIE INTEGER);

Таблица страны:

id\_str – индентификатор страны

nazvanie- это поле содержит название страны

stolica – в данном поленаходится столица выбранной страны

naselenie- население данной страны(в млн).

CREATE TABLE LOG (

"WHEN" TIMESTAMP,

"USER" VARCHAR(20),

"ACTION" CHAR(3));

Таблица лог: данная таблица содержит информацию о том Кто? Когда? и что сделал(удалил,изменил,вставил)?

When- дата дейстивя;

User- кто совершает действия;

Action- показывает действие которое совершает пользователь

Таблица страны:

id\_str – индентификатор страны

nazvanie- это поле содержит название страны

stolica – в данном поленаходится столица выбранной страны

naselenie- население данной страны(в млн).

CREATE TABLE LOG (

"WHEN" TIMESTAMP,

"USER" VARCHAR(20),

"ACTION" CHAR(3));

Таблица лог: данная таблица содержит информацию о том Кто? Когда? и что сделал(удалил,изменил,вставил)?

When- дата дейстивя;

User- кто совершает действия;

Action- показывает действие которое совершает пользователь

2.2 Создание триггеров

REATE TRIGGER TRIG\_AI\_GOR FOR GORODA

ACTIVE AFTER INSERT POSITION 0

AS

BEGIN

INSERT INTO log VALUES (current\_timestamp, current\_user, 'INS');

END

Данный триггер следит за действием вставка, если пользователь добавляет новое значение в таблицу goroda, триггер автоматически записывает в таблицу LOG имя пользователя и дату.

CREATE TRIGGER TR\_AD\_GOR FOR GORODA

ACTIVE AFTER DELETE POSITION 0

AS

BEGIN

INSERT INTO log VALUES (current\_timestamp, current\_user, 'DEL');

END

Данный триггер отвечает за удаление, если пользователь удалил значение из таблицы goroda, триггер автоматически записывает в таблицу LOG имя пользователя и дату.

CREATE TRIGGER TR\_AU\_GOR FOR GORODA

ACTIVE AFTER UPDATE POSITION 0

AS

BEGIN

INSERT INTO log VALUES (current\_timestamp, current\_user, 'UPD');

END

Данный триггер следит за изменением, если пользователь изменил любое поле из таблицы goroda, триггер автоматически записывает в таблицу LOG имя пользователя и дату.

2.3 Создание генераторов

CREATE GENERATOR GEN\_GORODA\_ID;

SET GENERATOR GEN\_GORODA\_ID TO 24;

CREATE GENERATOR GEN\_STRANA\_ID\_STR;

SET GENERATOR GEN\_STRANA\_ID\_STR TO 15;

Данные генераторы необходимы для создания триггеров на автоинкремент.

В них можно указать начальное значение автоинкремента.

CREATE TRIGGER TRIG\_GORODA FOR GORODA

ACTIVE BEFORE INSERT POSITION 0

as

begin

if(new.id\_gor is null) then new.id\_gor = gen\_id(gen\_goroda\_id,1);

end

Данный триггер автоматически создает идентификационный номер каждого нового города в таблице goroda.

CREATE TRIGGER TRIG\_STRANA FOR STRANA

ACTIVE BEFORE INSERT POSITION 0

AS BEGIN

IF (NEW.Id\_str IS NULL) THEN NEW.Id\_str = GEN\_ID(gen\_strana\_id\_str,1);

END

Тоже самое делает и ‘тот триггер. только для таблицы strana, отметим что идентификационный номер каждой новой страны или города увеличивается на 1.

2.4 Создание хранимых процедур

CREATE PROCEDURE STRANA\_1

RETURNS (

STRANA\_S VARCHAR(30),

GOROD\_G VARCHAR(30),

NASELENIE\_S INTEGER)

AS

begin

for select strana.nazvanie,goroda.nazvanie,strana.naselenie

from strana,goroda

where(strana.id\_str=goroda.id\_str)

into :strana\_s,:gorod\_g,:naselenie\_s

do suspend;

end

Данная процедура осуществляет выборку из таблиц Strana и Goroda.

В данном варианте при выполнении хранимой процедуры будут показаны название стран, городов, а так же население стран. Выборка осуществляется по идентификатору страны.

2.5 Разработка приложения для базы данных с помощью Borland Delphi

Основные компоненты которые использовались для создания приложения:

IBDatabase- Компонент TDatabase позволяет наиболее эффективно управлять подключениями к базам данных. Это управление включает:

- создание постоянного подключения к базе данных;

- отмену установленной по умолчанию регистрации на сервере;

- создание псевдонима BDE на уровне приложения;

- управление транзакциями и определение уровней их разграничения.

IBTransaction- компонент IBTransaction, который будет использоваться в качестве умолчательного для выполнения различных операций IBDatabase. Клиентская часть InterBase допускает выполнение любых действий только в контексте транзакции. Поэтому если вы смогли получить доступ к данным без явного вызова IBTransaction.StartTransaction, то значит где то в недрах IBX этот вызов произошел автоматически. Такое поведение крайне не рекомендуется использовать. Для корректной работы приложений с базой данных желательно управлять транзакциями вручную, то есть явно вызывать методы StartTransaction, Commit и Rollback компонента TIBTransaction.

IBDataSet- Назначение компонента: буферизация записей, выбираемых оператором SELECT, для представления этих данных в Grid, а также для обеспечения "редактируемости" записи путем автоматического или ручного задания запросов Insert, Delete и Update.

DataSource- Свойство DataSet обеспечивает связь между компонентом, представляющим собой таблицу или запрос, и компонентами, предназначенными для доступа к записям. Наличие этого свойства позволяет выбирать источник данных. Например, база данных может быть организована таким образом, что таблица, состоящая из большого числа записей, разделена на несколько подтаблиц, имеющих одинаковую структуру. В этом случае в приложении каждой подтаблице будет соответствовать свой компонент Table, а выбор конкретной подтаблицы можно осуществить установкой значения свойства DataSet.

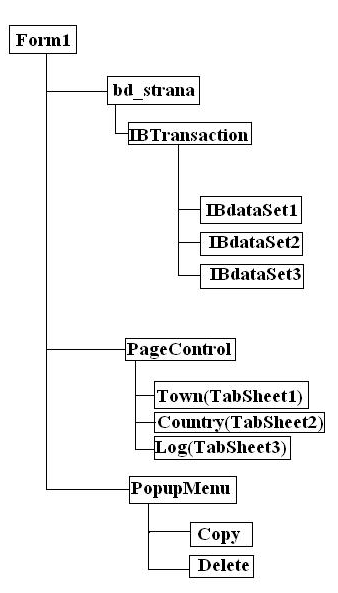
PageControl- Компонент Т PageControl может содержать несколько прикрывающих друг друга панелей класса TtabSheet. Каждая панель выбирается связанной с ней закладкой и может содержать свой набор помещенных на нее компонентов.

DBGrid- омпонент DBGrid обеспечивает представление базы данных в виде таблицы. Свойства компонента DBGridl определяют вид таблицы и действия, которые могут быть выполнены над данными во время работы программы.

PopupMenu- контекстное меню.

2.6 Блок схема

Рисунок. №1

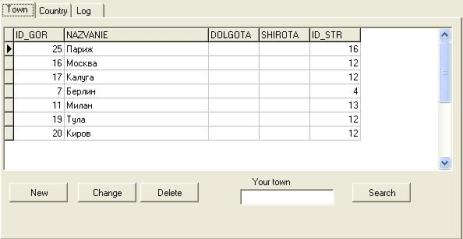


Глава 3. Руководство пользователю

Для того, чтобы открыть приложение базы данных необходимо двойным кликом мыши нажать на ярлык .

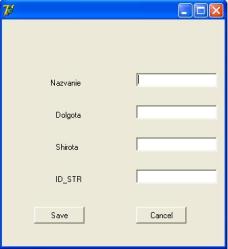
После запуска приложения появится основное окно программы

Рисунок. №2



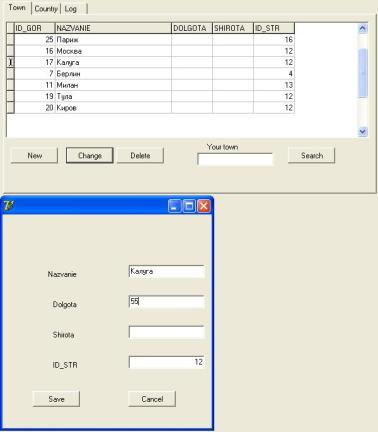
Для добавления новых данных необходимо нажать на кнопку New, после чего появится дополнительное окно в которое необходимо ввести новые данные.

Рисунок. №3



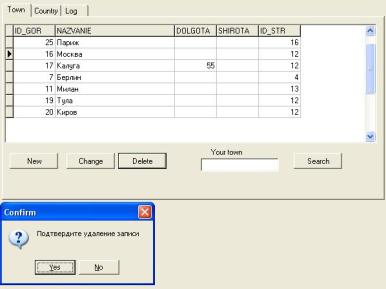
Для того чтобы изменить уже существующие данные, необходимо выбрать интересующую запись и нажать кнопку Change, после чего появится дополнительное окно в котором можно изменить данные. После внесенных изменений необходимо нажать кнопку Save для сохранения, либо Cancel для отмены.

Рисунок. №4



Для удаления существующих записей, необходимо выбрать ту запись, которую следует удалить и нажать кнопку Delete.

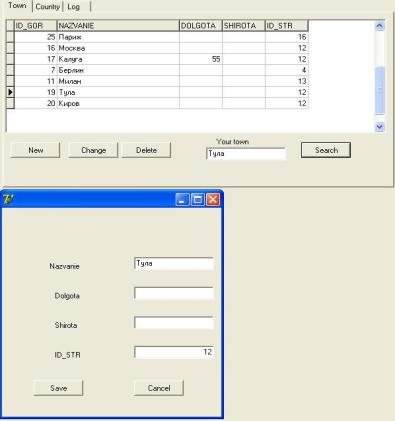
Рисунок. №5



Так же в базе данных существует поиск по названию страны(города).

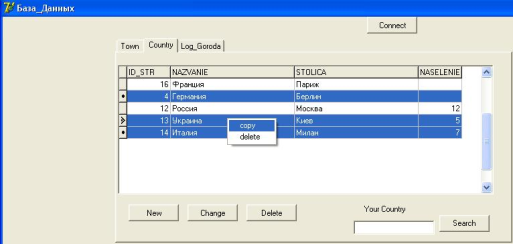
Для того, чтобы осуществить поиск, необходимо ввести название страны(города) в соответствующее поле и нажать кнопку Search.

Рисунок. №6



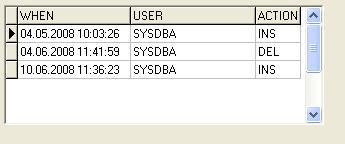
Для того чтобы скопировать необходимые данные из таблицы , нужно зажав кнопку Shift левой клавишей мыши, выбрать интересующие записи, после чего нажать правой кнопкой мыши для вызова контекстного меню и выбрать Copy.

Рисунок. №7



В базе данных ведется лог событий таблицы города. Все изменения отображаются в закладке Log\_Goroda.

Рисунок. №8



Заключение

В данной работе были кратко описаны основные характеристики системы управления базами данных. Определены основные классы объектов БД, которые имеют значение при использовании для решения наиболее распространенных задач. Interbase- хорошее решение для предприятий, стремящихся совершенствовать управление бизнесом в условиях постоянно изменяющегося рынка, стремящихся в максимально короткие сроки получить правильное решение.

Разработанное приложение осуществляет основные операции: добавление, изменение, удаление данных. Разработан поиск по названию стран (городов).Возможно копирование данных из таблицы. Создана лог-таблица, в которой ведется учет изменения данных.

Список литературы

1. www.ibase.ru;
2. Фаронов В.В. Delphi 6. Учебный курс. Москва 2003г.
3. Шупрута В.В. Delphi 2005.
4. Петр Дархвелидзе, Евгений Марков Программирование в Delphi7 СПб, 2003г.

5. megalib.com

Листинг программы

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, Grids, DBGrids, DB, IBCustomDataSet, IBDatabase, StdCtrls,

ComCtrls, Buttons, Mask, DBCtrls, Menus, QClipbrd;

type

TForm1 = class(TForm)

IBDatabase1: TIBDatabase;

IBTransaction1: TIBTransaction;

IBDataSet1: TIBDataSet;

DataSource1: TDataSource;

DataSource2: TDataSource;

IBDataSet2: TIBDataSet;

PageControl1: TPageControl;

TabSheet1: TTabSheet;

TabSheet2: TTabSheet;

DBGrid1: TDBGrid;

DBGrid2: TDBGrid;

BitBtn1: TBitBtn;

BitBtn2: TBitBtn;

BitBtn3: TBitBtn;

BitBtn4: TBitBtn;

BitBtn5: TBitBtn;

BitBtn6: TBitBtn;

BitBtn7: TBitBtn;

Edit1: TEdit;

Label1: TLabel;

Button1: TButton;

Edit2: TEdit;

Label2: TLabel;

BitBtn8: TBitBtn;

TabSheet3: TTabSheet;

DBGrid3: TDBGrid;

DataSource3: TDataSource;

IBDataSet3: TIBDataSet;

PopupMenu1: TPopupMenu;

copy1: TMenuItem;

delete1: TMenuItem;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

procedure BitBtn3Click(Sender: TObject);

procedure BitBtn2Click(Sender: TObject);

procedure BitBtn4Click(Sender: TObject);

procedure BitBtn5Click(Sender: TObject);

procedure BitBtn6Click(Sender: TObject);

procedure BitBtn7Click(Sender: TObject);

procedure BitBtn8Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure DBGrid1DblClick(Sender: TObject);

procedure DBGrid2DblClick(Sender: TObject);

procedure copy1Click(Sender: TObject);

private { Private declarations }

public { Public declarations }

end;

var

Form1: TForm1;

implementation

uses Unit2, Unit3, Unit4, unit5;

{$R \*.dfm}

procedure TForm1.Button1Click(Sender: TObject);

begin

if ibdataset2.Locate('NAZVANIE',Edit2.Text,[loPartialKey]) then form2.Show

else

showmessage('Не найдено');

//form1.ibdataset2.insert;

//form2.show;

end;

procedure TForm1.Button2Click(Sender: TObject);

begin

form1.IBDataSet1.Post;

end;

procedure TForm1.BitBtn1Click(Sender: TObject);

begin

ibdataset1.Edit;

//ibdataset1.Post;

form3.Show;

end;

procedure TForm1.BitBtn3Click(Sender: TObject);

begin

form1.IBDataSet1.Insert;

//form1.IBDataSet1.Post;

form3.Show;

end;

procedure TForm1.BitBtn2Click(Sender: TObject);

begin

ibdataset2.Insert;

//ibdataset2.Post;

form2.Show;

end;

procedure TForm1.BitBtn4Click(Sender: TObject);

begin

if MessageDlg ('Подтвердите удаление записи',

mtConfirmation, [mbYes, mbNo], 0) = mrYes then

ibdataset1.Delete

end;

procedure TForm1.BitBtn5Click(Sender: TObject);

begin

ibdataset2.Edit;

form2.Show;

end;

procedure TForm1.BitBtn6Click(Sender: TObject);

begin

if MessageDlg ('Подтвердите удаление записи',

mtConfirmation, [mbYes, mbNo], 0) = mrYes then

ibdataset2.Delete;

end;

procedure TForm1.BitBtn7Click(Sender: TObject);

begin

if ibdataset1.Locate('NAZVANIE',Edit1.Text,[loPartialKey]) then form3.Show

else

showmessage('Не найдено');

end;

procedure TForm1.BitBtn8Click(Sender: TObject);

begin

Form4.show;

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

ibdataset1.Open;

ibdataset2.Open;

end;

procedure TForm1.DBGrid1DblClick(Sender: TObject);

begin

form5.Show;

form5.IBDataSet1.Open;

end;

procedure TForm1.DBGrid2DblClick(Sender: TObject);

begin

form5.Show;

form5.IBDataSet1.Open;

end;

procedure TForm1.copy1Click(Sender: TObject);

var

i, j: Integer;

s: string;

begin

if DBGrid2.SelectedRows.Count>0 then

with DBGrid2.DataSource.DataSet do

for i:=0 to DBGrid2.SelectedRows.Count-1 do

begin

GotoBookmark(pointer(DBGrid2.SelectedRows.Items[i]));

for j := 0 to FieldCount-1 do

begin

if (j>0) then s:=s+', ';

s:=s+Fields[j].AsString;

end;

s:= s+#13+#10;

end;

Clipboard.AsText := s;

end;

end.

unit Unit2;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, DBCtrls, Mask, Buttons;

type

TForm2 = class(TForm)

DBEdit2: TDBEdit;

DBEdit3: TDBEdit;

DBEdit4: TDBEdit;

Button1: TButton;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

BitBtn1: TBitBtn;

procedure Button1Click(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form2: TForm2;

implementation

uses unit1;

{$R \*.dfm}

procedure TForm2.Button1Click(Sender: TObject);

begin

form1.IBDataSet2.Post;

form2.Close;

end;

procedure TForm2.BitBtn1Click(Sender: TObject);

begin

form2.Close;

end;

end.

unit Unit3;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, Buttons, Mask, DBCtrls;

type

TForm3 = class(TForm)

DBEdit2: TDBEdit;

DBEdit3: TDBEdit;

DBEdit4: TDBEdit;

DBEdit5: TDBEdit;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

BitBtn1: TBitBtn;

BitBtn2: TBitBtn;

procedure BitBtn1Click(Sender: TObject);

procedure BitBtn2Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form3: TForm3;

implementation

uses unit1;

{$R \*.dfm}

procedure TForm3.BitBtn1Click(Sender: TObject);

begin

form1.IBDataSet1.Post;

form3.Close;

end;

procedure TForm3.BitBtn2Click(Sender: TObject);

begin

form3.Close;

end;

procedure TForm3.FormCreate(Sender: TObject);

begin

end;

end.

unit Unit4;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls;

type

TForm4 = class(TForm)

Edit1: TEdit;

Edit2: TEdit;

Button1: TButton;

procedure Button1Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form4: TForm4;

implementation

uses unit1; {$R \*.dfm}

procedure TForm4.Button1Click(Sender: TObject);

begin

Form1.IBDatabase1.Params.Clear;

Form1.IBDatabase1.Params.Add('user\_name='+Edit1.Text);

Form1.IBDatabase1.Params.Add('password='+Edit2.Text);

Form1.IBDatabase1.Params.Add('lc\_ctype=WIN1251');

Form1.IBDatabase1.Connected:=True;

Form1.IBTransaction1.StartTransaction;

Form1.IBDataSet1.Open;

Form1.IBDataSet2.Open;

Form1.IBDataSet3.Open;

form4.Close;

end;

procedure TForm4.FormCreate(Sender: TObject);

begin

end;

end.

unit Unit5;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, DB, IBCustomDataSet, Grids, DBGrids;

type

TForm5 = class(TForm)

DBGrid1: TDBGrid;

DataSource1: TDataSource;

IBDataSet1: TIBDataSet;

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form5: TForm5;

implementation

uses unit1;

{$R \*.dfm}

procedure TForm5.FormCreate(Sender: TObject);

begin

end.