Министерство образования и науки Украины

Севастопольский национальный технический университет

ИЗУЧЕНИЕ ОСНОВ ЯЗЫКА МАНИПУЛИРОВАНИЯ ДАННЫМИ SQL НА БАЗЕ СЕРВЕРА INTERBASE

Методические указания

к лабораторной работе №1

по дисциплине

“ОРГАНИЗАЦИЯ БАЗ ДАННЫХ”

для студентов специальности 7.080401 –

"Информационные управляющие системы и технологии"

всех форм обучения

Севастополь
2005

УДК 004.92

Изучение основ языка манипулирования данными SQL на базе сервера Interbase. Методические указания к лабораторной работе №1 по дисциплине “Организация баз данных”, для студентов специальности 7.080401 -"Информационные управляющие системы и технологии" всех форм обучения. /Сост. Ю.В. Доронина, О.Л. Тимофеева, В.Е. Шишкевич. - Севастополь: Изд-во СевНТУ, 2005.-20с.

Цель методических указаний: выработка у учащихся практических навыков по работе с реляционными базами данных.

Методические указания рассмотрены и утверждены на заседании кафедры Информационных Систем. Протокол № 7 от 19 мая 2005 г.

Рецензент: доц. департамента кибернетики и вычислительной техники, к.т.н. Литвинова Л.А.

Допущено учебно-методическим центром в качестве методических указаний.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В связи с широким внедрением вычислительной техники во все области деятельности человека, включая и повседневную жизнь, с одной стороны, и все увеличивающиеся потоки информации, с другой стороны, в настоящее время большую популярность получили различные электронные справочники, информационно - поисковые системы. В большинстве случаев, такие системы по своим размерам являются малыми и средними базами данных.

При разработке таких систем применяется теория реляционных баз данных, использующая сложный математический аппарат. Проектированию реляционных БД посвящено множество монографий, учебников и статей.

Данные методические указания предназначены для формирования навыков работы с распределенными реляционными БД. Предложенный комплекс лабораторных работ включает базовые принципы и подходы, которые необходимы для организации работы с современными средствами хранения информации. Так в лабораторной работе №1 рассматриваются правила организации работы в среде многопользовательской реляционной БД Interbase. Лабораторная работа №2 – основные операторы языка описания данных и языка манипулирования данными.
Лабораторные работы № 3,5,6,7 - базовые операции по манипулированию данными, отображение реляционных операций на языке манипулирования данными SQL, простые и сложные (вложенные) запросы на выборку данных. Лабораторные работы №4 - требования к проектированию набора отношений реляционной БД для обеспечения целостности (достоверности) данных. Лабораторная работа №8 – средства для автоматического формирования значения ключа вводимых данных. Лабораторная работа №9 – способы создания автоматически вызываемых процедур, обеспечивающих дополнительные условия поддержания целостности данных при выполнении операций, изменяющих хранимые данные.

ЛАБОРАТОРНАЯ РАБОТА № 1.
ИЗУЧЕНИЕ АРХИТЕКТУРЫ СЕРВЕРА INTERBASE

1. Цель работы:

Изучить основы организации сервера Interbase; научится устанавливать соединение с сервером. Научиться создать базу данных и производить элементарные действия над ней.

2. Основные положения

2.1.Описание сервера

Сервер Interbase фирмы Borland является полнофункциональным сервером реляционных баз данных. Начиная с версии 6.0, Borland сделала доступным исходный текст сервера для бесплатного использования. На основе исходного текста IB 6.0 появилось несколько клонов IB под разными именами. Сторонние разработчики исправляли ошибки, выявленные в исходном коде, и расширяли функциональность сервера. Одним из самых известных клонов стал FireBird разработкой и сопровождением, которого занимается сообщество разработчиков, включая бывших сотрудников подразделения Borland Interbase.

Сервер выпускается под руководством несколько платформ (Win32, Linux, Solaris, freebsd, Darwin) в двух архитектурах – Super Server и Classic. Основное различие между ними состоит в том, что Classic создает параллельный процесс для каждого присоединяемого пользователя, а SuperServer состоит из одного процесса, который обрабатывает запросы клиентов в разных нитях (threads) этого же процесса. Архитектура Classic считается более надежной, а SuperServer более производительной. Для платформы Win32 сервер выпускается только в архитектуре SuperServer.

Сервер на Win32 запускается в качестве сервиса ОС. Клиентские приложения могут присоединяться к нему несколькими способами: по протоколам NetBEUI, TCP-IP; локальное подключение (в случае, если вы работаете на машине, на которой запущен сервер). В дальнейшем рассматривается подключение к серверу по протоколу TCP-IP.

Основной утилитой для работы с сервером будет IBConsole, она входит в поставку сервера. Для того чтобы получить возможность работы с БД, необходимо проделать следующие операции:

- Зарегистрировать сервер
- Присоединиться к серверу
- Зарегистрировать (или создать) базу данных
- Присоединиться к базе данных

На каждой машине регистрировать сервер и БД нужно только один раз.

2.2. Регистрация сервера

На рисунке 1 изображен вид экрана пользователя после запуска, утилиты IB-Console в котором не зарегистрировано ни одного сервера.

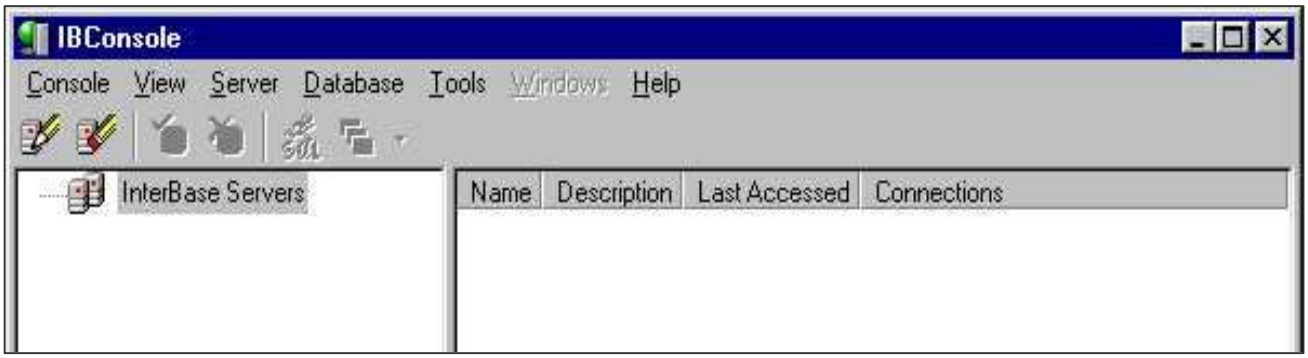

Рисунок 1 – Вид IBConsole без зарегистрированных БД

Для регистрации сервера служит команда Server->Register. Появится диалоговое окно Register server and connect. Нужно указать следующую информацию:

- сервер удаленный (Remote Server);
- имя сервера (Server Name) – ISIBServer. Имя сервера есть имя машины, на котором установлен Firebird. Имя сервера необходимо для определения IP-адреса машины, на которой установлен сервер. Преобразованием имени в IP-адрес в разных сетях занимаются разные службы. Это может быть WINS-сервер в сетях Microsoft, DNS-сервер в сетях IP. Выполним преобразование средствами Windows. Для этого в системном каталоге Windows нужно создать текстовый файл с именем hosts (или отредактировать его, если он существует), и добавить в него следующую строку: “ISIBServer 10.0.50.101”. Первая часть строки содержит текстовое имя, затем, через пробел, следует IP-адрес, ассоциируемый с именем. Проверить, возможно, ли соединение с сервером, можно набрав в командной строке DOS (Far) команду ping ISIBServer;
- Сетевой протокол (Network protocol) – TCP/IP;
- Псевдоним сервера (Alias name) – строка, под которой сервер будет виден в IBConsole. Введите IBServer;
- Поясняющий текст (Description) – можно оставить пустым;
- Имя пользователя (User Name) – stud;
- Пароль (Password) – isthebest;

Правильно заполненное диалоговое окно представлено на рисунке 2. На рисунке 3 изображен вид IBConsole после успешной регистрации и соединения с сервером.

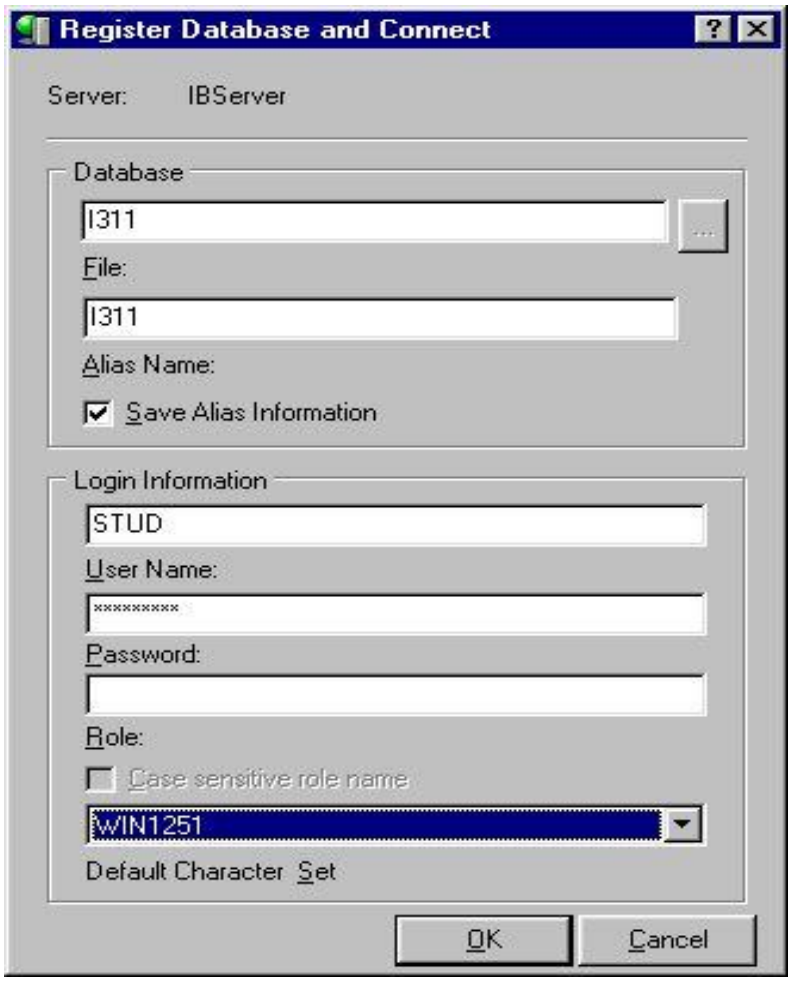

Рисунок 2 – Окно регистрации сервера

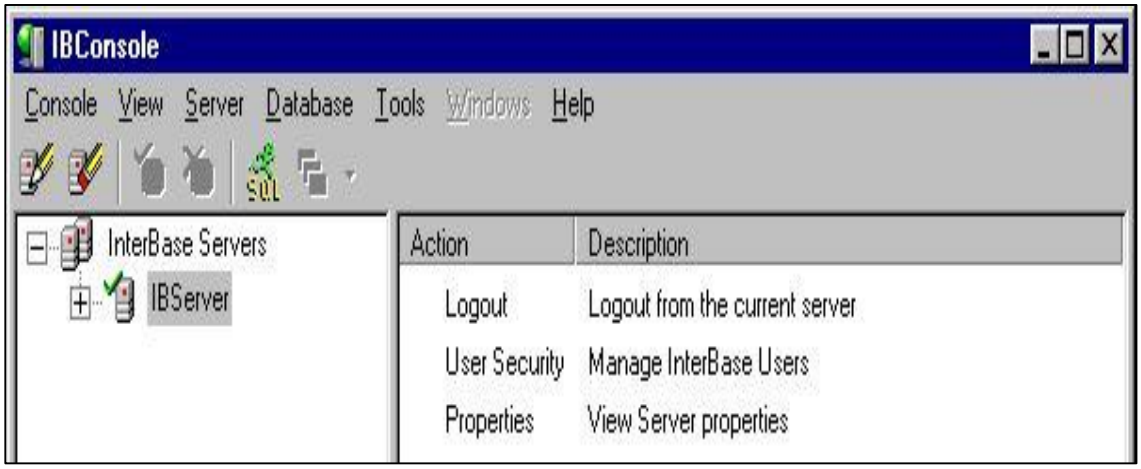

Рисунок 3 – Сервер, с которым установлено соединение, отмечен зеленой галочкой

2.3. Присоединение к серверу

Диалоговое окно регистрации сервера автоматически производит подключение к серверу. В дальнейшем подключение к серверу можно осуществить по двойному нажатию левой кнопки мыши на имени сервера. На рисунке 4 изображено диалоговое окно Server Login, с помощью которого можно подключиться к серверу, пользователю необходимо указать логин и пароль аналогично п.2.2 регистрация сервера.

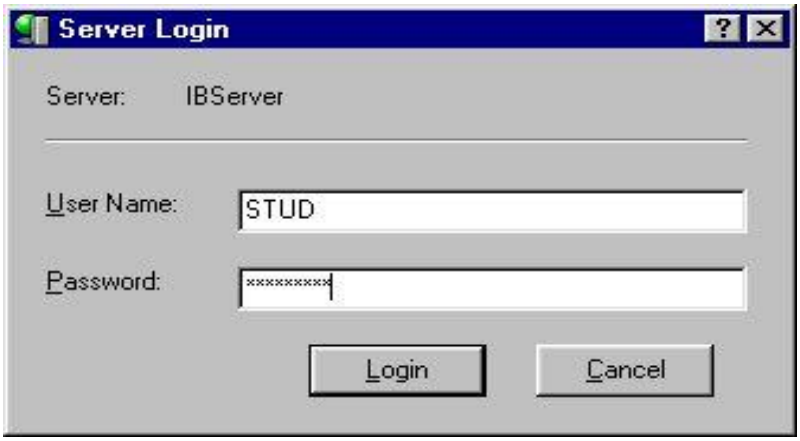

Рисунок 4 – Окно присоединения к серверу

2.4. Регистрация и создание базы данных

После соединения с сервером, необходимо выбрать команду Database->Create Database. Появится диалоговое окно создания базы данных. Необходимо указать следующую информацию:

- из каких файлов будет состоять база данных. Очень большие БД могут состоять из нескольких файлов, хранящихся на разных дисках. Учебная БД будет состоять из одного файла. Имя файла составляется из Номера группы и номера студента по списку группы (либо номера подгруппы в группе).
- Псевдоним БД – это имя, под которым БД показывается в IBConsole, псевдоним повторяет имя файла БД.

На рисунке 5 изображен вид диалогового окна с правильно заполненными полями данных. После создания автоматически происходит присоединение к базе данных. На рисунке 6 изображен вид IBConsole после присоединения к БД.

Необходимо обратить внимание на вкладки Domains, Tables, Views и др. В дальнейшем они будут содержать создаваемые объекты базы данных – домены, таблицы, пользовательские отображения (View), хранимые процедуры и т.д.

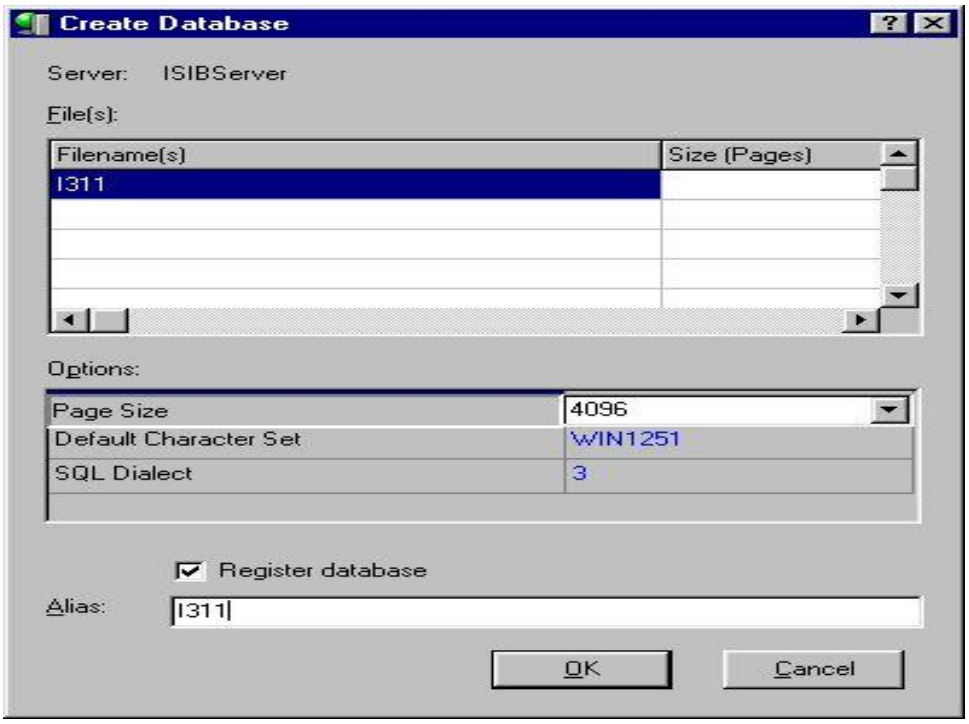

Рисунок 5 – Окно создания БД

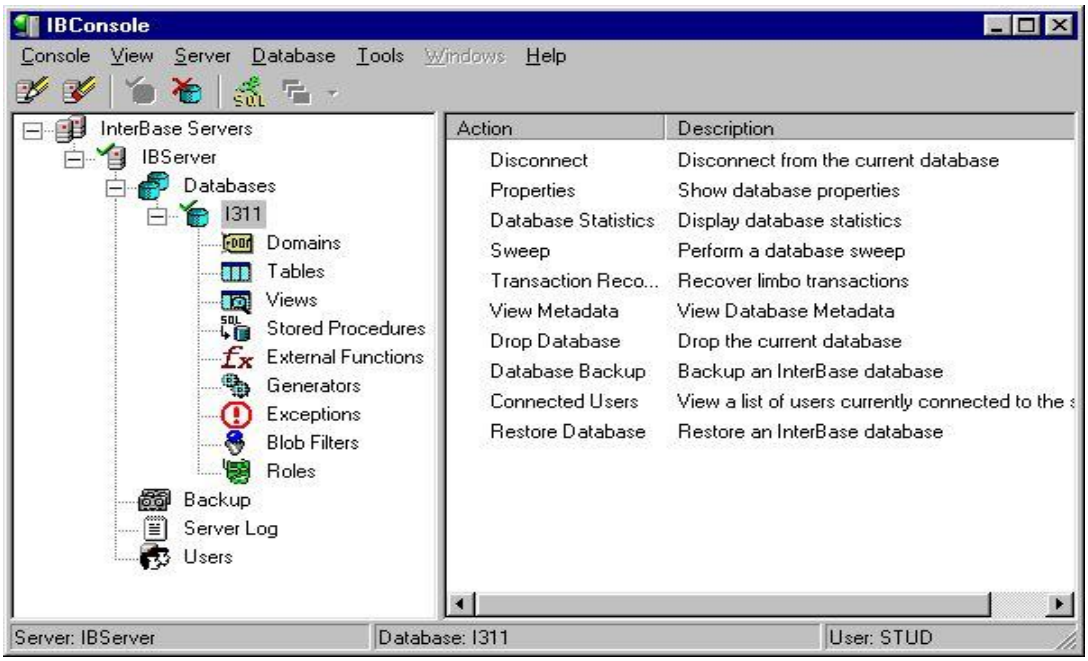

Рисунок 6 – Вид IBConsole после присоединения к БД

Регистрация уже существующей базы данных производится командой Database->Register. Появится диалоговое окно регистрации базы данных, которое представлено на рисунке 7. Необходимо заполнить следующие поля:

- имя файла базы данных (File) – в точности то же самое, что было указано при создании.

- Псевдоним БД – тот же, что и при создании БД.
- Обязательно надо установить набор символов по умолчанию (Default character set) в WIN1251.

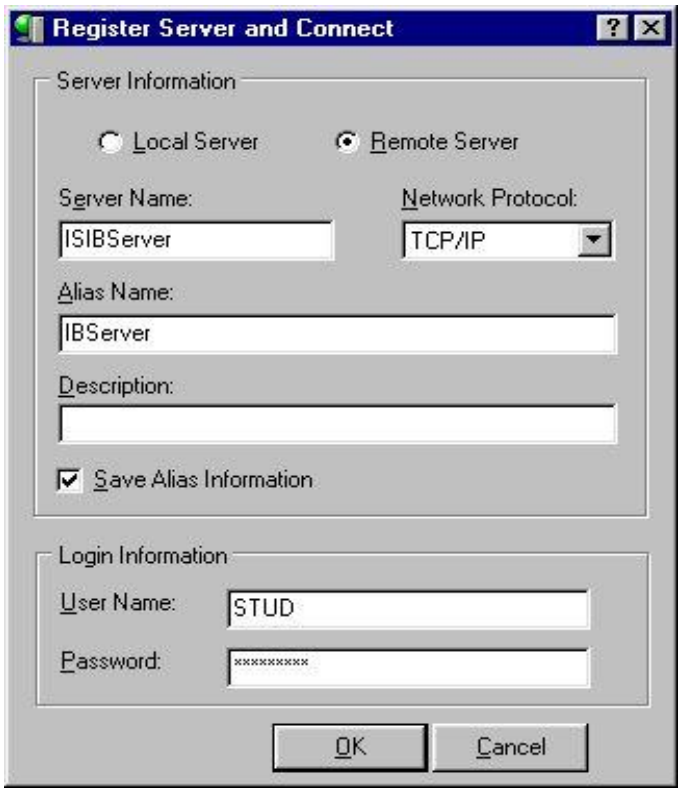

Рисунок 7 – Диалоговое окно регистрации БД

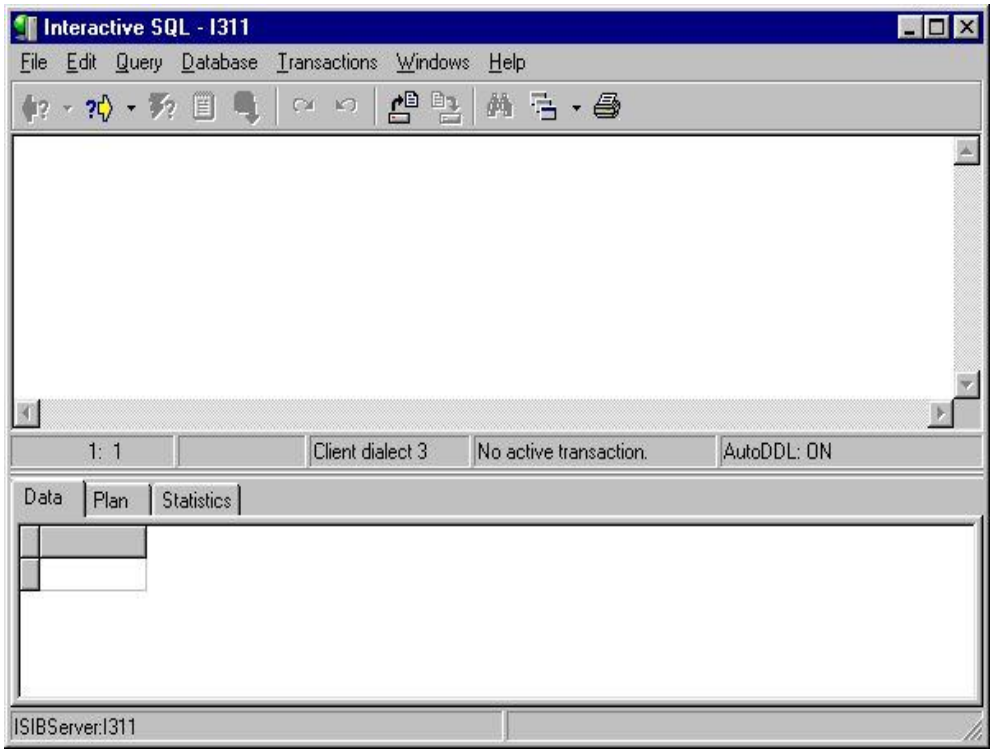

Рисунок 8 – Утилита Interactive SQL

После регистрации сервера и создания БД можно приступать к работе с базой. Для этого служит утилита Interactive SQL (ISQL), вызываемая из IBConsole командой Tools-> Interactive SQL. На рисунке 8 изображен вид рабочего окна утилиты.

Рабочее окно утилиты ISQL разбито на две части. В верхней области можно вводить команды SQL, внизу отображается результат выполнения команды, план, созданный оптимизатором для запроса, и статистика выполнения запроса.

2.5. Элементарные операции с БД

Создание таблицу с именем test, состоящую из двух полей, одно из которых целочисленное, второе – строковое.

Create table **test(** col1 **integer,** col2 **varchar(20))**

Добавление строки в таблицу

insert into test **values(**1,'Vasya'**)**

Просмотр содержимого таблицы.

select * from test

Удаление таблицы

drop table test

Последняя команда вызовет ошибку «Object in use» - объект используется. Это произошло потому, что вся работа с ISQL осуществляется в контексте транзакции, т.е. все изменения не будут внесены в базу данных до тех пор, пока их не подтвердят.

Для подтверждения изменений служит команда Transactions->Commit, для отмены - Transactions->Rollback.

Только после подтверждения транзакции сервер создаст таблицу и добавит строку данных. После этого можно снова попытаться удалить таблицу, удаление пройдет успешно.

3. Ход работы

1. Ознакомиться с описанием организации работы реляционных БД.
2. Установить соединение с сервером БД.
3. Создать тестовую БД, в соответствии с п.2.4, произвести регистрацию созданной БД.
4. В соответствии, с п.2.5:
   - создать тестовую таблицу,
   - занести в таблицу пять кортежей,
   - просмотреть содержимое таблицы,
   - удалить созданную таблицу.

4. Содержание отчета

1. Отчет должен содержать описание действий студента по конкретному варианту.

2. Обязательно должны быть указаны фактические значения, вводимые в диалоговые окна.
3. Отчет состоит из титульного листа, цели работы, описания процесса выполнения работы и вывода.

5. Контрольные вопросы

1. Объясните отличие архитектуры superserver и classic?
2. Какая информация указывается при регистрации базы данных?
3. Базовое понятие многопользовательских систем «транзакция»?
4. Напишите запрос, добавляющий в таблицу «test» только второе поле?
5. Объясните порядок регистрации сервера?
6. Модели данных. Реляционная модель данных?
7. СУБД – назначение?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Боуман Д.С. Практическое руководство по SQL. Использование диалектов SQL./Д.С.Боуман. –М.;СПб.;К.: Вильямс, 2001.- 351 с.
2. Боуман Д.С. Практическое руководство по SQL. Использование языка структурированных запросов./ Д.С.Боуман –М.; СПб.;К.: Вильямс, 2001.- 334 с.
3. Елланд Ф.Д. Основі концепции баз данніх./Ф.Д. Елланд. –М.; СПб.;К.: Вильямс, 2002.- 235 с.
4. Конноли Т. Базы данных. Проектирование, реализация и сопровождение теория и практика./Т.Конноли. –М.; СПб.;К.: Вильямс, 2001.- 1111 с.
5. Кузнецов С. SQL. Язык реляционных Баз данных./С. Кузнецов. –М.: Майор, 2001. -191 с.
6. Ульман Дж.Д Введение в системы баз данных./Дж.Д. Ульман.–М.:Лори, 2000. - 374 с.

ПРИЛОЖЕНИЕ А
(обязательное)
SQL.

Синтаксис оператора CREATE TABLE

```
CREATE TABLE table [EXTERNAL [FILE] "<filespec>"]
(<col_def> [, <col_def> | <tconstraint> ...]);
<col_def> = col {datatype | COMPUTED [BY] (<expr>) | domain}
 [DEFAULT {literal | NULL | USER}]
  [NOT NULL] [<col_constraint>]
  [COLLATE collation]
```

Здесь

COMPUTED [**BY**] – “Вычисляется как”

DEFAULT – значение по умолчанию

[<ограничение столбца>] – ограничение целостности столбца

Обратите внимание: Предложение COLLATE не может быть определено для столбцов BLOB.

Имя столбца должно быть обязательно указано. Далее должен быть указан тип данных столбца или выражение для его вычисления, или имя домена. Опционально могут быть указаны:

- значение по умолчанию
- ограничения целостности (PRIMARY KEY, UNIQUE, FOREIGN KEY)
- атрибут NOT NULL (обязательно должен быть указан у PRIMARY KEY и FOREIGN KEY)
- условие CHECK – условие, которое должно быть истинным для того, чтобы строка могла быть добавлена к таблице

Описание конструкций оператора CREATE TABLE представлено в таблице А.1.

Таблица А.1. – Описание конструкций оператора CREATE TABLE

Аргумент	Описание
Table	Имя для таблицы, должно быть уникальным среди имен таблиц и процедур в базе данных.
EXTERNAL [FILE] "<filespec>"	Объявляет что данные для создаваемой таблицы, постоянно располагаются во внешней к базе данных таблице или файле.
Col	Имя для столбца таблицы, должно быть уникальным именем в таблице.
<datatype>	SQL тип данных столбца.
COMPUTED [BY] (<expr>)	Указывает, что значение столбца вычисляется по другим столбцам. Выражение должно возвращать одиночное значение, и не иметь тип массива или возвращать массив.
<expr> -	любое арифметическое выражение допустимое для типа данных столбца.
domain	Имя существующего домена.

Продолжение таблицы А.1.

COLLATE collation	Определяет порядок сортировки для столбца. Порядок сортировки на уровне столбца отменяет порядок сортировки определенный на уроне домена
DEFAULT	Определяет значение по умолчанию столбца, если оно не определяется каким-либо другим образом Значения: - literal: Вставляется указанная строка, числовое значение или дата. - NULL: Вводится значение NULL - USER: Вводится имя текущего пользователя. Столбец должен иметь тип, совместимый с текстовым Установка значению по умолчанию на уровне столбца отменяет значение по умолчанию на уровне домена
CONSTRAINT constraint	Помещает именованное ограничение на таблицу или столбец. Если имя ограничения опущено, InterBase создает системное имя для ограничения.

Ограничение целостности CHECK

Синтаксис:

CHECK *(*<search_condition>*);*

В условии можно пользоваться операторами сравнения, операторами NOT, BETWEEN, LIKE, IN, IS [NOT] NULL и прочими. У столбца может быть только одно условие CHECK.

Условие (<search_condition>) должно принимать значение истинно, для того чтобы разрешить операцию добавления или изменения данных. <search_condition> может проверять несколько значений, введенных в другие столбцы. Это ограничение на уровне таблицы.

Допустимы следующие типы данных

<datatype> =

{SMALLINT | INTEGER | FLOAT | DOUBLE PRECISION}[<array_dim>] | (DATE | TIME | TIMESTAMP}[<array_dim>] | {DECIMAL | NUMERIC} [(precision [, scale])] [<array_dim>] | {CHAR | CHARACTER | CHARACTER VARYING | VARCHAR} [(int)] [<array_dim>] [CHARACTER SET charname] | {NCHAR | NATIONAL CHARACTER | NATIONAL CHAR} [VARYING] [(int)] [<array_dim>] | BLOB [SUB_TYPE {int | subtype_name}] [SEGMENT SIZE int] [CHARACTER SET charname]
| BLOB [(seglen [, subtype])]<array_dim> = [[x:]y [, [x:]y …]]

Синтаксис оператора INSERT

INSERT INTO <object> [(col [, col ...])]
{VALUES (<val> [, <val> ...]) | <select_expr>};
<object> = tablename | viewname
<val> = {<constant> | <expr>| <function> | NULL | USER} [COLLATE collation]

Обратите внимание: Предложение COLLATE не может быть использовано для BLOB значений.

Описание конструкций оператора INSERT представлено в таблице А.2.

Таблица А.2 – Описание конструкций оператора INSERT

Аргумент	Описание
INTO <object>	Имя существующей таблицы или вида, в которую вставляются данные.
Col	Имя существующего столбца в таблице или представлении, в который вставляются значения.
VALUES (<val> [, <val> ...]	Список значению для вставки в таблицу или представление. Значения должны быть в том же порядке, как целевые столбцы.
<select_expr>	Запрос, который возвращает значения, для вставки в целевые столбцы.

<constant> = num | "string" | charsetname "string"

<expr> = Допустимое выражение SQL, которое возвращает в одиночное значение столбца.

<function> = {CAST (<val> AS <datatype>)
| UPPER (<val>)
| GEN_ID (generator, <val>) }

<select_expr> = SELECT возвращающий ноль или более строк, где число столбцов в каждой строке такое же, как число элементов, которые должны быть вставлены.

Синтаксис оператора SELECT

```
SELECT [DISTINCT | ALL] {* | <val> [, <val> ...]}
 FROM <tableref> [, <tableref> ...]
  [WHERE <search_condition>]
   [GROUP BY col [COLLATE collation] [, col [COLLATE collation] ...]
                [HAVING <search_condition>]
   [UNION <select_expr>]
   [PLAN <plan_expr>]
   [ORDER BY <order_list>]
```

Описание конструкций оператора SELECT представлено в таблице А.3.

Таблица А.3 – Описание конструкций оператора SELECT

Аргумент	Описание
SELECT [DISTINCT | ALL]	Определяет данные, для поиска. DISTINCT удаляет повторяющиеся значения из возвращенных данных. ALL, параметр по умолчанию, возвращает все данные.
*	возвращает все столбцы из определенной таблицы.
<val>[,<val> ...]	возвращает определенный список столбцов и значений

Продолжение таблицы А.3.

FROM <tableref> [, <tableref> ...]	Список таблиц, представлений и сохраненных процедур, из которых возвращаются данные. Список может включать JOIN, соединения могут быть вложенными.
Table	Имя существующей таблицы в базе данных.
View	имя существующего представления в базе данных.
Procedure	Имя существующей сохраненной процедуры, которая функционирует, как инструкция SELECT.
Alias	Псевдоним - краткое, альтернативное имя для таблицы или представления.
<joined_table>	Ссылка таблицы состоящая из JOIN.
<join_type>	Тип выполнения соединения. По умолчанию INNER.
WHERE <search_cond>	Определяет условия, которые ограничивают подмножество возвращаемых строк из всех доступных строк.
GROUP BY <col>[, <col> ...]	Разделяет результаты запроса в группы содержащие все строки с одинаковыми значениями, основанными на списке столбцов.
COLLATE collation	Определяет порядок сопоставления для данных возвращаемых запросом.
HIVING <search_cond>	Используется совместно с GRUP BY. Определяет условия, которые ограничивают группировку возвращаемых строк.
UNION	Комбинирует две или более таблиц, которые имеют полностью, либо частично одинаковую структуру.
PLAN <plan_expr>	Определяет план доступа для оптимизатора INTERBASE,который используется в течении поиска.
<plan_item>	определяет таблицу и индексный метод для плана.
ORDER BY <order_list>	Определяет порядок в котором строки будут возвращены.

Оператор ALTER TABLE

Используется для модификации структуры существующей таблицы

Удаление столбца

ALTER TABLE <имя таблицы> DROP <имя столбца>[, <имя столбца>...];
Например:
ALTER TABLE EMPLOYEE DROP EMP_NUM, JOB_CODE;

Команда ALTER TABLE не сможет выполниться при следующих условиях:

- если данные в таблице после удаления нарушают ограничения PRIMARY KEY или UNIQUE
- если столбец является частью первичного ключа, ограничения UNIQUE, или столбец является внешним ключом
- столбец используется в ограничении CHECK
- столбец используется в представлении (view), в триггере, или используется при вычислении другого поля.

В случае, если что-либо мешает удалить столбец, надо сначала удалить это «что-то», а затем удалять столбец.

Добавление столбца

ALTER TABLE <имя таблицы> ADD <определение столбца>

Например:

ALTER TABLE employee ADD emp_no INTEGER NOT NULL;

Добавление ограничения целостности на таблицу

ALTER TABLE <имя таблицы> ADD [CONSTRAINT <имя ограничения>]
<определение ограничения>;

Можно добавить ограничения PRIMARY KEY, FOREIGN KEY, UNIQUE, или CHECK.

Например:

ALTER TABLE EMPLOYEE ADD CONSTRAINT DEPT_NO UNIQUE (PHONE_EXT);

Удаление ограничения целостности, наложенного на столбец

ALTER TABLE <имя таблицы>
DROP CONSTRAINT <имя ограничения>;

Например:

ALTER TABLE PROJECT DROP CONSTRAINT TEAM_CONSTRT;

Если имя ограничения не было указано явно при создании таблицы, сервер генерирует имя автоматически. Посмотреть имена ограничений можно в системной таблице RDB$RELATION_CONSTRAINTS:

SELECT * FROM RDB$RELATION_CONSTRAINTS

Модификация столбца в таблице:

Можно изменить имя столбца, тип столбца, позицию столбца в таблице.

ALTER TABLE <имя таблицы> ALTER [COLUMN] <имя столбца> <определение изменения>

<определение изменеия>:

- для имзменеия имени: TO <новое имя>
- для изменеия типа данных: TYPE <новый тип>
- для изменения позиции: POSITION <целое число>

Примеры:

1. Изменение позиции
   ALTER TABLE EMPLOYEE ALTER EMP_NUM POSITION 2;

2. Изменение имени EMP_NUM на EMP_NO
   ALTER TABLE EMPLOYEE ALTER EMP_NUM TO EMP_NO;

3. Изменение типа
   ALTER TABLE EMPLOYEE ALTER EMP_NO TYPE CHAR(20);

Преобразование из несимвольных в символьные типы возможны при следующих ограничениях.

- BLOB и массивы не преобразуются.
- Новое определение поля должно вмещать старые данные (CHAR(3) не вместит CHAR(4))