САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ФАКУЛЬТУТ ТЕХНИЧЕСКОЙ КИБЕРНЕТИКИ

М.Ю. Моисеев, В.А. Цесько, А.В. Мяснов

ОСНОВЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ И ИНФОРМАЦИОННЫХ СИСТЕМ

Методические указания к лабораторному практикуму

Санкт-Петербург
2012

УДК 004.6
ББК 32.973-018.2

В пособии приведено краткое описание содержания лабораторных работ, выполняемых на кафедре компьютерных систем и программных технологий Санкт-Петербургского государственного политехнического университета. Представлены задания и последовательность выполняемых действий. Приведены ссылки на интернет-ресурсы, которые рекомендованы в качестве справочного материала для выполнения работ.

Пособие предназначено для студентов, обучающихся по направлениям «Информатика и вычислительная техника» и «Автоматизация и управление» и призвано помочь студентам в закреплении знаний по дисциплинам «Базы данных» и «Информационное обеспечение систем управления».

Содержание

1. Вводная работа

Цели работы

Познакомить студентов с организацией лабораторных работ и используемыми средствами для выполнения работ (получения заданий и необходимой информации, разработки, хранения исходных кодов и отчетов, представления результатов работ). Изучить основные сущности реляционной БД и их назначение, дать начальные навыки создания простейшей БД.

Программа работы

1. Ознакомление с организацией работ
2. Изучение работы с БД с помощью `isql`
3. Самостоятельное изучение основ языка SQL
4. Выбор темы работы
5. Создание БД для полученного задания. Необходимо создать 2 таблицы, определить в каждой из них первичный ключ. Для одной из таблиц определить ограничение уникальности значений в поле, создать вторичный индекс.
6. Согласно полученному заданию определить какая из таблиц является главной, придумать и реализовать связь таблиц с помощью ограничения внешнего ключа
7. Заполнить таблицы осмысленными данными (5-10 записей)
8. Проверить работу введенных ограничений. Для ограничений внешнего ключа проверить различные варианты действий при удалении/изменении записи в главной таблице.
9. Продемонстрировать все результаты преподавателю
10. Обсудить тему с преподавателем, согласовать перечень сущностей информационной системы
11. Отчёт должен содержать ER-диаграмму разрабатываемой БД

Утилита `isql` находится в каталоге `D:\Progam Files\Firebird\Firebird_2_0\bin\`.

Для создания новой БД с помощью `isql` необходимо использовать команду **create database**, указав путь к БД, имя пользователя, пароль, например:

```
$ isql.exe
> create database
'tiger.ftk.spbstu.ru:/var/lib/firebird/<группа>/<имя_базы>.fdb' user 'SYSDBA'
password 'masterkey';
> commit;
```

Для подключения к БД необходимо использовать команду **connect**, например:

```
> connect 'tiger.ftk.spbstu.ru:/var/lib/firebird/<группа>/<имя_базы>.fdb'
user 'SYSDBA' password 'masterkey';
```

Для удаления БД нужно присоединиться к БД используя команду **connect** и использовать команду **drop database**, например:

```
> drop database;
> commit;
```

При создании базы данных, таблиц и других сущностей, необходимо использовать команду **commit** перед их использованием.

Полезные ссылки

- Курс "Введение в базы данных"
- Модификация полей таблицы, добавление ограничений
- Firebird 2.1 Language Reference Update
- Описание SQL на русском языке
- Развёрнутая документация по хранимым процедурам в СУБД Firebird
- Справочник SQL
- Entity-relationship Model

Основные сущности СУБД Firebird

Реляционную БД можно рассматривать как набор таблиц, хотя в базу данных могут входить также процедуры и ряд других объектов. Таблицу в БД можно представлять как обычную двумерную таблицу с характеристиками (атрибутами) какого-то множества объектов. Таблица имеет имя, по которому на нее можно сослаться. Столбцы таблицы соответствуют тем или иным характеристикам объектов - полям. Каждое поле характеризуется именем и типом хранящихся данных. Каждая строка таблицы соответствует одному из объектов. Она называется записью и содержит значения всех полей, характеризующие данный объект.

При добавлении объектов в БД обычно требуется обеспечить уникальность и непротиворечивость информации. Обычно это делается введением ключевых полей, обеспечивающих уникальность каждой записи - это поле или поля называются первичным ключом.

При работе с таблицей пользователь или программа перемещает курсор по записям таблицы. В каждый момент времени есть некоторая текущая запись, с которой и ведется работа. Записи в таблице базы данных физически могут располагаться без какого-либо порядка, просто в последовательности их добавления. При выборке (представлении) часто требуется упорядочить данные некоторым образом. Для упорядочивания данных используются **индексы**. Индекс определяет, в какой последовательности будут выдаваться строки таблицы. Если индекс включает в себя несколько полей, то упорядочивание базы данных сначала осуществляется по первому полю, а для записей, имеющих одинаковые значения первого поля - по второму и т.д. Для первичного ключа всегда создается индекс. Кроме этого можно создать **вторичные индексы.**

База данных обычно содержит более одной таблицы. При обработке данных часто требуется одновременно выбирать данные из нескольких логически связанных таблиц. В реляционной БД организуются связи двух таблиц между собой следующим образом : одна таблица является главной, другая - подчиненной, таблицы связываются друг с другом с помощью **внешнего ключа**. Внешний ключ определяется полем или набором полей в главной таблице и полем или набором полей в подчиненной таблице. Количество и типы полей внешнего ключа должны совпадать в главной и подчиненной таблицах. Поле или набор полей главной таблицы должны быть уникальными, обычно в главной таблице выбирается поле или поля являющиеся первичный ключом. Каждой записи в главной таблице внешний ключ ставит в соответствие, в общем случае, множество записей подчиненной таблицы, таким образом, реализуется **связь один ко многим.**

Часто первичный ключ и внешний ключ называют **ограничениями**. Это связано с тем, что наличие ключей ограничивает данные, которые можно занести в таблицу. Так первичный ключ требует, чтобы в таблице все записи были уникальными. Наличие первичного ключа хотя и необязательно, но обычно каждая таблица имеет первичный ключ. Часто в качестве первичного ключа вместо выбора поля или набора полей-атрибутов объекта создается специальное поле, обычно типа `Integer`. Такой первичный ключ называется **суррогатным**, он заполняется последовательными значениями начиная с 0 (обычно автоматически при добавлении записи). Внешний ключ таже накладывает ограничения на данные в главной и подчиненной таблицах - для каждой записи в подчиненной таблице должна существовать ровно одна запись в главной таблице с таким значением внешнего ключа. Другими вида ограничений являются ограничения уникальности для выбранного поля и ограничения на диапазон вводимых в поле значений, в том числе и на допустимость пустого значения (`NULL`).

Для ограничения значений, которые может принимать некоторый атрибут можно создавать пользовательские типы данных - **домены**. Домен можно создавать на основе базовых типов данных.

2. Разработка структуры и нормализация БД

Цели работы

Познакомить студентов с основами проектирования схемы БД, способами нормализации отношений в БД.

Программа работы

1. Представить схему БД, соответствующую заданию (должно получиться не менее 7 таблиц)
2. Привести схему БД к 3НФ
3. Согласовать с преподавателем схему БД. Обосновать соответствие схемы 3НФ.
4. Продемонстрировать результаты преподавателю.

Материалы работы

- Описание SQL на русском языке: http://www.mstu.edu.ru/education/materials/zelenkov/ch_4_6_1.html

3. SQL-DDL

Цели работы

Познакомить студентов с основами проектирования схемы БД, языком описания сущностей и ограничений БД SQL-DDL.

Программа работы

1. Самостоятельное изучение SQL-DDL
2. Создание скрипта БД в соответствии с согласованной схемой (должны присутствовать первичные и внешние ключи, ограничения на диапазоны значений). Продемонстрировать скрипт преподавателю.
3. Создайте скрипт, заполняющий все таблицы БД данными
4. Выполнение SQL-запросов, изменяющих схему созданной БД по заданию преподавателя. Продемонстрировать их работу преподавателю.
5. Изучите основные возможности IBExpert. Получите ER-диаграмму созданной БД с помощью **Database Designer**.
6. Автоматически сгенерируйте данные при помощи IBExpert (для трех или большего числа таблиц, не менее 100000 записей)

Язык SQL

Язык SQL (Structured Query Language) - язык структурированных запросов. Он позволяет формировать весьма сложные запросы к базам данных. В SQL определены два подмножества языка:

- SQL-DDL (Data Definition Language) - язык определения структур и ограничений целостности баз данных. Сюда относятся команды создания и удаления баз данных; создания, изменения и удаления таблиц; управления пользователями и т.д.
- SQL-DML (Data Manipulation Language) - язык манипулирования данными: добавление, изменение, удаление и извлечение данных, управления транзакциями

Типы данных

Символьные типы данных:

- `CHAR(n)` - символьные строки фиксированной длины. Длина строки определяется параметром n. CHAR без параметра соответствует CHAR(1). Для хранения таких данных всегда отводится n байт вне зависимости от реальной длины строки.
- `VARCHAR(n)` - символьная строка переменной длины. Для хранения данных этого типа отводится число байт, соответствующее реальной длине строки.

Целые типы данных:

- `SMALLINT` - короткое целое (2 байта)
- `INTEGER` - обычное целое (4 байта)
- `BIGINT` - длинное целое (8 байт)

Вещественные типы данных:

- `FLOAT` - числа с плавающей точкой (4 байта).
- `DOUBLE PRESCISION` - числа с плавающей точкой (8 байт).
- `DECIMAL(p,n)` - тип данных аналогичный FLOAT с числом значащих цифр p и точностью n.

Дата и время - используются для хранения даты, времени и их комбинаций:

- `DATE` - тип данных для хранения даты.
- `TIME` - тип данных для хранения времени.
- `TIMESTAMP` - тип данных для хранения моментов времени (год + месяц + день + часы + минуты + секунды + доли секунд).

Двоичные типы данных - позволяют хранить данные любого объема в двоичном коде (оцифрованные изображения, исполняемые файлы и т.д.):

- `BLOB`

Описание SQL

- Interbase 6 Language Reference
- Firebird 2.1 Language Reference Update
- Основные объекты SQL-программирования (ХП, триггеры, представления, ...): SQL-DDL?
- Описание SQL на русском языке: http://www.mstu.edu.ru/education/materials/zelenkov/ch_4_6_1.html
- Развёрнутая документация по хранимым процедурам в СУБД Firebird
- Справочник SQL
- Утилиты Firebird, PSQL

4. Язык SQL-DML

Цели работы

Познакомить студентов с языком создания запросов управления данными SQL-DML.

Программа работы

1. Изучите SQL-DML
2. Выполните все запросы из списка стандартных запросов. Продемонстрируйте результаты преподавателю.
3. Получите у преподавателя и реализуйте SQL-запросы в соответствии с **индивидуальным** заданием. Продемонстрируйте результаты преподавателю.
4. Выполненные запросы `SELECT` сохраните в БД в виде представлений, запросы `INSERT`, `UPDATE` или `DELETE` -- в виде ХП. Выложите скрипт в Subversion.

Список стандартных запросов

- Сделайте выборку всех данных из каждой таблицы
- Сделайте выборку данных из одной таблицы при нескольких условиях, с использованием логических операций, `LIKE`, `BETWEEN`, `IN` (не менее 3-х разных примеров)
- Создайте в запросе вычисляемое поле
- Сделайте выборку всех данных с сортировкой по нескольким полям
- Создайте запрос, вычисляющий несколько совокупных характеристик таблиц
- Сделайте выборку данных из связанных таблиц (не менее двух примеров)
- Создайте запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки
- Придумайте и реализуйте пример использования вложенного запроса
- С помощью оператора `INSERT` добавьте в каждую таблицу по одной записи
- С помощью оператора `UPDATE` измените значения нескольких полей у всех записей, отвечающих заданному условию
- С помощью оператора `DELETE` удалите запись, имеющую максимальное (минимальное) значение некоторой совокупной характеристики
- С помощью оператора `DELETE` удалите записи в главной таблице, на которые не ссылается подчиненная таблица (используя вложенный запрос)

Язык SQL

Язык SQL (Structured Query Language) -- язык структурированных запросов. Он позволяет формировать весьма сложные запросы к базам данных. В SQL определены два подмножества языка:

- SQL-DDL (Data Definition Language) -- язык определения структур и ограничений целостности баз данных. Сюда относятся команды создания и удаления баз данных; создания, изменения и удаления таблиц; управления пользователями и т.д.
- SQL-DML (Data Manipulation Language) -- язык манипулирования данными: добавление, изменение, удаление и извлечение данных, управления транзакциями

Типы данных

Символьные типы данных:

- `CHAR(n)` -- символьные строки фиксированной длины. Длина строки определяется параметром `n`. `CHAR` без параметра соответствует `CHAR(1)`. Для хранения таких данных всегда отводится `n` байт вне зависимости от реальной длины строки.
- `VARCHAR(n)` -- символьная строка переменной длины. Для хранения данных этого типа отводится число байт, соответствующее реальной длине строки.

Целые типы данных:

- `SMALLINT` -- короткое целое (2 байта)
- `INTEGER` -- обычное целое (4 байта)
- `BIGINT` -- длинное целое (8 байт)

Вещественные типы данных:

- `FLOAT` -- числа с плавающей точкой (4 байта)
- `DOUBLE PRECISION` -- числа с плавающей точкой (8 байт)
- `DECIMAL(p,n)` -- тип данных аналогичный `FLOAT` с числом значащих цифр `p` и точностью `n`

Дата и время -- используются для хранения даты, времени и их комбинаций:

- `DATE` -- тип данных для хранения даты
- `TIME` -- тип данных для хранения времени
- `TIMESTAMP` -- тип данных для хранения моментов времени (год + месяц + день + часы + минуты + секунды + доли секунд)

Двоичные типы данных - позволяют хранить данные любого объема в двоичном коде (оцифрованные изображения, исполняемые файлы и т.д.):

- `BLOB`

Оператор выбора `SELECT`

ВАЖНО: В запросах для строковых литералов используются одинарные кавычки (`'строка'`), двойные кавычки используются для имён полей и таблиц (`"поле"`).

Для извлечения записей из таблиц в SQL определен оператор `SELECT`. Этот оператор возвращает ни одного, одно или множество строк, удовлетворяющих указанному условию и упорядоченных по заданному критерию. Это наиболее сложное и мощное средство SQL, полный синтаксис оператора `SELECT` имеет вид:

```
SELECT [ALL | DISTINCT] <список_выбора>
FROM <имя_таблицы>, ...
[ WHERE <условие> ]
[ GROUP BY <имя_столбца>,... ]
[ HAVING <условие> ]
[ ORDER BY <имя_столбца> [ASC | DESC],... ]
```

В квадратных скобках приведены необязательные части оператора SELECT. Порядок предложений в операторе SELECT должен строго соблюдаться:

- SELECT <список_выбора> -- поля таблицы значения из которых нужно выбрать (* - дает все поля таблицы)
- FROM <имя_таблицы> -- имя таблицы, из которой выбираются данные
- WHERE <условие> -- условие отбора (выбираются только записи, которые ему соответствуют)
- GROUP BY <имя_столбца>[, <имя_столбца> ...] -- определяет условие группировки
- HAVING <условие> -- накладывает ограничения на результаты группировки
- ORDER BY <имя_столбца> -- определяет по каким полям будут упорядочиваться возвращаемые записи
- ASC | DESC -- указывают, как сортировать по возрастанию или по убыванию

Совокупные характеристики

Оператор SELECT позволяет возвращать не только множество значений полей, но и некоторые совокупные (агрегированные) характеристики, подсчитанные по всем или по указанным записям таблицы. В SQL определены следующие агрегатные функции:

- AVG (<имя поля>) -- среднее по всем значениям данного поля
- SUM (<имя поля>) -- сумма всех значений данного поля
- MAX (<имя поля>) -- максимальное из всех значений данного поля
- MIN (<имя поля>) -- минимальное из всех значений данного поля
- COUNT (<имя поля>) или COUNT (*) -- число записей

Оператор INSERT

Добавление записей осуществляется с помощью оператора INSERT:

```
INSERT INTO <имя_таблицы> [ (<имя_поля>, ...) ] VALUES (<значение>, ...)
```

Производится добавление одной записи в указанную таблицу <имя_таблицы> и заносятся значения для выбранных полей. Если имена полей не указаны, то значения заносятся во все поля таблицы. Набор полей должен быть согласован с набором заносимых значений. Для остальных полей заносятся значения определенные по умолчанию, а если таковых не имеется то в них записывается NULL.

Оператор DELETE

Удаление одной или нескольких записей осуществляется с помощью оператора DELETE:

```
DELETE FROM <имя_таблицы> [ WHERE <условие> ]
```

Удаляются все записи, удовлетворяющие указанному условию. Если ключевое слово WHERE и условие отсутствуют, из таблицы удаляются все записи.

Оператор UPDATE

Модификация записей осуществляется оператором UPDATE:

```
UPDATE <имя_таблицы>
SET <имя_столбца> = <значение>, ...
[WHERE <условие>]
```

Для заданной таблицы изменяются значения указанных в операторе столбцов. Если задано ключевое слово `WHERE` и условие, то оператор `UPDATE` применяется только к тем записям, для которых оно выполняется. Если условие не задано, `UPDATE` применяется ко всем записям.

Представления

Представления -- виртуальные именованные таблицы, создаваемые с помощью запроса `SELECT`.

Описание SQL

- Interbase 6 Language Reference
- Firebird 2.1 Language Reference Update
- Описание SQL на русском языке: http://www.mstu.edu.ru/education/materials/zelenkov/ch_4_6_1.html
- Развёрнутая документация по хранимым процедурам в СУБД Firebird
- Справочник SQL

5. SQL-программирование: хранимые процедуры

Цели работы

Познакомить студентов с возможностями реализации более сложной обработки данных на стороне сервера с помощью хранимых процедур.

Программа работы

1. Изучить возможности языка `PSQL`
2. Создать две хранимые процедуры в соответствии с индивидуальным заданием, полученным у преподавателя
3. Выложить скрипт с созданными сущностями в svn
4. Продемонстрировать результаты преподавателю

Хранимые процедуры

Хранимые процедуры -- части программы, находящиеся и выполняющиеся на сервере. Они хранятся вместе с базой данных на сервере в откомпилированном виде и позволяют перенести часто повторяющиеся длительные операции с клиентской машины на SQL сервер.

Материалы

- Introducing the InterBase Procedure and Trigger Language
- Развернутая документация по хранимым процедурам.
- Firebird 2.1 Language Reference Update
- Interbase 6 Language Reference

6. SQL-программирование:триггеры, вызовы процедур

Цели работы

Познакомить студентов с возможностями реализации более сложной обработки данных на стороне сервера с помощью хранимых процедур и триггеров.

Программа работы

1. Создать два триггера: один триггер для автоматического заполнения ключевого поля, второй триггер для контроля целостности данных в подчиненной таблице при удалении/изменении записей в главной таблице
2. Создать триггер в соответствии с индивидуальным заданием, полученным у преподавателя
3. Создать триггер в соответствии с индивидуальным заданием, вызывающий хранимую процедуру
4. Выложить скрипт с созданными сущностями в svn
5. Продемонстрировать результаты преподавателю

Триггеры

Триггеры - программы обработчики определенных событий, хранящиеся на сервере. Обрабатываемые события это добавление, изменение или удаление данных определенной таблицы, т.е. триггер связан с некоторой таблицей.

Материалы

- Introducing the InterBase Procedure and Trigger Language
- Развернутая документация по хранимым процедурам.
- Firebird 2.1 Language Reference Update
- Interbase 6 Language Reference

7. Разработка клиентского приложения

Варианты приложения

1. Консольное приложение
    - импорт и экспорт в xml нескольких таблиц,
    - проверка с помощью схемы.
2. Графическое приложение с CRUD (технология Swing) для 3-4 таблиц
    - master-detail с редактированием,
    - отчет с использованием запроса,
    - запуск процедур и получение результатов.
3. Веб-приложение на JSP
    - master-detail с редактированием,
    - отчет с использованием запроса,
    - запуск процедур и получение результатов.
4. Веб-приложение на XML + XSLT
    - master-detail,
    - отчет с использованием запроса,
    - запуск процедур и получение результатов.

8. Использование Hibernate

Цель работы

Целью данной работы является ознакомление с концепцией объектно-реляционного отображения.

Программа работы

1. Ознакомиться с технологией Hibernate.
2. Создайте новое приложение. Реализуйте объектно-реляционное отображение.
3. Реализуйте всю функциональность приложения.
4. Протестируйте приложение в многопользовательском режиме (для этого нужно запустить два или более экземпляров приложения).
5. Проанализируйте достоинства и недостатки изученной технологии.

9. Изучение механизма транзакций

Цель работы

Познакомить студентов с механизмом транзакций, возможностями ручного управления транзакциями, уровнями изоляции транзакций.

Программа работы

1. Изучить основные принципы работы транзакций.
2. Провести эксперименты по запуску, подтверждению и откату транзакций.
3. Разобраться с уровнями изоляции транзакций в Firebird.
4. Спланировать и провести эксперименты, показывающие основные возможности транзакций с различным уровнем изоляции.
5. Продемонстрировать результаты преподавателю, ответить на контрольные вопросы.

Работа проводится в IBExpert. Для проведения экспериментов параллельно запускается несколько сессий связи с БД, в каждой сессии настраивается уровень изоляции транзакций. Выполняются конкурентные операции чтения/изменения данных в различных сессиях, а том числе приводящие к конфликтам.

Документация

- Немного о транзакциях
- Уровни изоляции
- Синтаксис `SET TRANSACTION`
- Опции транзакций

10. Оптимизация запросов

Цель работы

В рамках данной работы студенты знакомятся с методами повышения производительности запросов.

Программа работы

1. Изучить основные способы повышения производительности запросов
2. Написать SQL-запрос для индивидуального задания, полученного у преподавателя
3. С помощью средств IBExpert оценить производительность написанного SQL-запроса
4. Предложить варианты улучшения производительности с использованием:

    - ручного формирования плана выполнения запроса,
    - использования индексов,
    - денормализации БД.

5. Для каждого из вариантов провести эксперименты и проанализировать полученные результаты.
6. Написать отчет по проделанной работе.