Санкт-Петербургский политехнический университет

Кафедра компьютерных систем и программных технологий

**Отчёт по вводной лабораторной работе**

**Дисциплина**: Базы данных

**Тема**: Разработка клиентского приложения.

Выполнил студент гр. 43501/4 Чеботарёв Г.М.

(подпись)

Руководитель Мяснов А.В.

(подпись)

Санкт -Петербург

2016

**Программа работы**

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

**Ход работы:**

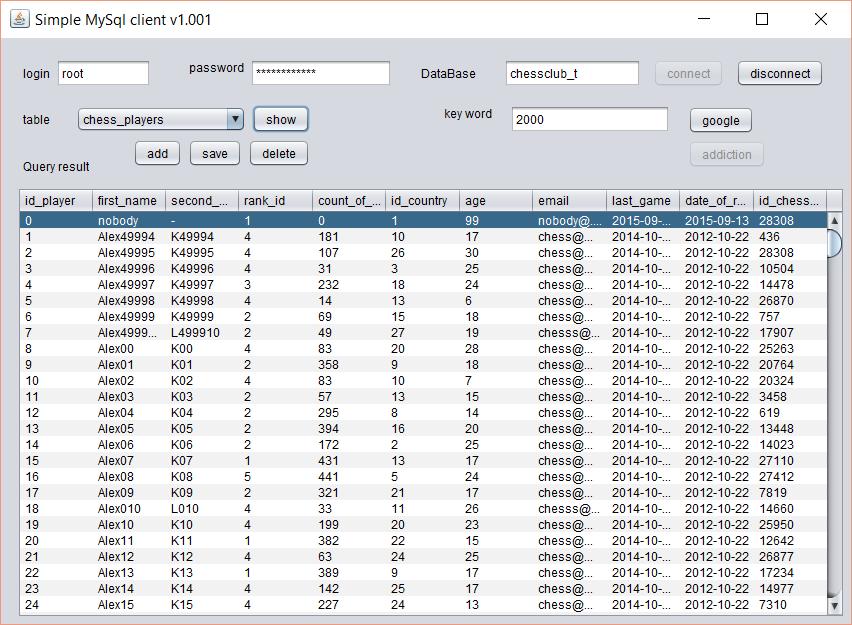


Рис. 1. – вид оконного приложения

Данное приложение написано на java с использованием JDBC библиотек. Реализована следующая функциональность:

1. Подключение (переподключение) к некоторой БД.
2. Возможность просматривать (изменять, добавлять, удалять) содержимое таблиц (любой таблицы из выбранной БД).
3. Возможность поиска ключевого слова в выбранной таблице, а также по таблицам, связанными внешними ключами с искомой, с последующим просмотром содержимого поля внешней таблицы.
4. Автоматическая подгрузка данных при прокрутке.
5. Система оповещения об ошибках.

Разработанное клиентское приложение протестировано на нескольких БД с различным содержимым. По результатам тестов, «комфортная работа» с таблицей до 130 Мб. При работе с таблицами 200-400 Мб возникают некоторые «торможения». Это обуславливается ограниченным потенциалом JDBC библиотеки.

***Запросы генерируются автоматически, согласно заполняемым полям:***

1. **Запрос данных на отображение**

String query = "select \* from " + jComboBox1.getSelectedItem() + " LIMIT "+start\_lim+","+Limit+";";

где ***jComboBox1.getSelectedItem()*** – имя таблицы, введенное пользователем в соответствующую графу,

***Limit*** – кол-во строк вытягиваемых за запрос.

1. **Запрос на добавление строки в таблицу:**

String query = "INSERT INTO `" + jComboBox1.getSelectedItem() + "` ( ", name;

for(int i =0; i< count\_header; i++){

query = query + (String)headerVect.get(i);

if(i+1<count\_header)

query = query +", ";

}

query = query +") VALUES ( ";

for(int i =0; i< count\_header; i++){

query = query + "'"+(String)new\_data.get(i)+"'";

if(i+1<count\_header)

query = query +" , ";

}

query = query +");";

boolean rs = stmt.execute(query);

В первом цикле генерируются заголовки, и конкатенируются. Во втором – добавляемые данные, взятые из jTable (т.е. то, что ввел пользователь в таблицу).

1. **Запрос на обновление данных**

String query = "update `" + jComboBox1.getSelectedItem() + "` set ", name;

for(int i =0; i< count\_header; i++){

query = query + (String)headerVect.get(i)+" = '"+(String)new\_data.get(i)+"'";

if(i+1<count\_header)

query = query +", ";

}

query = query +" where ";

for(int i =0; i< count\_header; i++){

query = query + (String)headerVect.get(i)+" = '"+(String)old\_data.get(i)+"'";

if(i+1<count\_header)

query = query +" and ";

}

query = query +";";

Как и в предыдущем запросе: первый цикл – генерируют данные которые нужно заменить, второй – генерирует условия замены.

1. **Запрос на удаление строки**

String query = "DELETE FROM `" + jComboBox1.getSelectedItem() + "` WHERE ", name;

for(int i =0; i< count\_header; i++){

query = query + (String)headerVect.get(i)+" = '"+(String) jTable1.getValueAt(index\_deleted,i)+"'";

if(i+1<count\_header)

query = query +" and ";

}

query = query +";";

Запрос содержит ряд переменных:

jComboBox1.getSelectedItem() – имя таблицы,

headerVect – вектор заголовков таблица,

jTable1.getValueAt(index\_deleted,i) – данные из конкретной ячейки

1. **Запрос на внешние ключи выбранной таблицы**

query = "SELECT COLUMN\_NAME as master\_col, REFERENCED\_TABLE\_NAME as name\_slave, "

+ "REFERENCED\_COLUMN\_NAME as slave\_col" +

" FROM information\_schema.KEY\_COLUMN\_USAGE\n" +

"WHERE TABLE\_SCHEMA ='"+jTextField3.getText()+"' AND TABLE\_NAME ='"+

jComboBox1.getSelectedItem() +"' AND\n" +

"CONSTRAINT\_NAME <>'PRIMARY' AND REFERENCED\_TABLE\_NAME is not null;";

Данный запрос возвращает все внешние ключи выбранной таблицы.

1. ***Генерация составного запроса для поиска ключевого слова по выбранной таблице, и таблицам, связанных с ней с помощью внешних ключей.***
   1. ***Поиск по основной таблице***:

query = "select \*, 'KEYTABLE', 'KEYWORD' from "+jComboBox1.getSelectedItem()+" where concat("

+ header\_slave\_table+") like '%"+jTextField2.getText()+"%' LIMIT 0,"+Limit+" ";

Основные переменные:

***jComboBox1.getSelectedItem()*** – имя текущие таблицы;

***header\_slave\_table*** – строка, в которой конкатенированы заголовки таблицы через запятую;

***jTextField2.getText()*** – ключевое слово введенное пользователем;

***Limit*** – кол-во строк вытягиваемых за запрос.

* 1. ***Поиск по второстипенним таблицам***

query = query + " UNION select T1.\*,'"+external\_keys\_vec.get(i+1)+"', "

+ "concat\_ws('=','"+external\_keys\_vec.get(i+2)+"',T2."+external\_keys\_vec.get(i+2)+

") from "+jComboBox1.getSelectedItem()+ " as T1\n" +

"inner join " + external\_keys\_vec.get(i+1)+ " as T2 on concat"

+ "(" + header\_slave\_table + ") like '%"+jTextField2.getText()+"%'\n" +

"where T1."+external\_keys\_vec.get(i)+" = T2."+external\_keys\_vec.get(i+2) + " LIMIT 0,"+Limit;

***external\_keys\_vec.get(i+2)*** – имя внешней таблицы;

***external\_keys\_vec.get(i+1)*** – имя поля во внешней таблице;

***jComboBox1.getSelectedItem()*** – имя текущие таблицы;

***header\_slave\_table*** – строка, в которой конкатенированы заголовки таблицы через запятую;

***jTextField2.getText()*** – ключевое слово введенное пользователем;

***Limit*** – кол-во строк вытягиваемых за запрос.

Подобные запросы генерируются для каждой из внешних таблиц, после чего «объединяются» с помощью UNION и отправляются на сервер mysql. Так как внешних ключей у таблицы не больше 3( chess\_players), подобная архитектура запроса допускается. В случае, если количество таблиц, к которым необходимо было бы послать было бы велико, предпочтительнее было бы отправлять запросы по отдельности. Полный код приложения выложен на GitHub, ссылка добавлена в комментарии к заданию.

**Вывод:**

В разработанном приложении реализованы следующие требования:

1. CRUD
2. Подгрузка при прокрутке
3. Поиск ключевого слова по выбранной таблице и связанных таблицах.

К положительным сторонам приложения можно отнести ее независимость от БД. Приложение будет одинаково работать как с chessclub\_t, так и с любой другой базой данных. При разработке были получены (возможно)полезные навыки. К недостаткам – полная непрактичность и ненадобность.