А.А. Петросян, С.А. Бурлов

МЕХАНИЗМ ЗАЩИЩЕННОГО ДОСТУПА К БАЗЕ ДАННЫХ ПРИ РАЗРАБОТКЕ ИНФОРМАЦОННОЙ СИСТЕМЫ

(Самарский национальный исследовательский университет
имени академика С.П. Королёва)

Введение. Большинство информационных систем имеют сложную архитектуру и, зачастую, над одним проектом работает группа разработчиков, каждый из которых отвечает за определенную часть системы. Такой способ работы значительно упрощает и ускоряет процесс разработки, но, в то же время, появляется проблема контроля безопасности и согласованности между разработчиками.

На стадии проекта, когда физическая модель данных сформирована, разработчики приступают к работе над серверным приложением. Подобная работа может также производиться после реализации и внедрения информационной системы. Предоставление разработчикам серверного приложения возможности самостоятельно формировать запросы к данным порождает следующие проблемы:

- отсутствие единого способа обращения к данным, синтаксическая разрозненность логически одинаковых запросов, как следствие, ухудшение читаемости исходного кода, нарушение инкапсуляции;
- ответственность за предотвращение *SQL*-инъекций лежит исключительно на плечах разработчиков серверного приложения;
- отсутствие возможности детального ограничения доступа к данным, формирования запросов к заданным таблицам или полям и только к ним, вне зависимости от схемы.

Результаты. Чтобы предотвратить появление описанных проблем, предлагается использовать процедурное расширение языка *SQL*. Вместо предоставления разработчикам серверного приложения возможности формировать запросы самостоятельно, администраторами базы данных формируются и предоставляются хранимые процедуры, выполняющие необходимые действия с данными, создаются публичные синонимы для того, чтобы исключить необходимость использования имени схемы, в которой расположена процедура. Разработчики серверного приложения должны иметь права на выполнение лишь предоставленных процедур, которые в полной мере реализуют необходимые процессы извлечения и загрузки данных, при этом возможность сформировать собственные запросы отсутствует.

Таким образом, разработчики серверного приложения получают единый интерфейс для работы с базой данных, предоставляющий возможность выполнения лишь определенных запросов посредством вызова хранимых процедур и функций. При попытке выполнить те же запросы, что содержатся в процедурах,

напрямую, выдаётся ошибка об отсутствии прав на операцию. Так же и другие запросы будут недоступны (Рисунок 1).

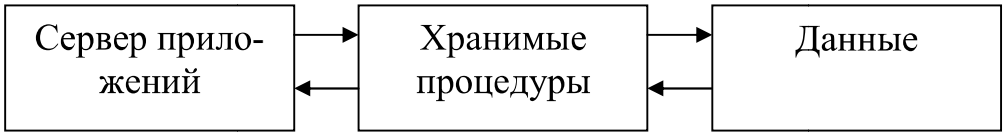

Рис. 1 – Схема взаимодействия с данными

Для выборки данных из таблицы администратор систем-источников данных предоставляет хранимую процедуру, одним из параметров которой должен быть курсор. В теле процедуры переданный курсор открывается для заданного администратором базы данных запроса. В качестве дополнительных входных параметров передаются опции запроса. Разработчик серверного приложения вызывает процедуру и получает открытый курсор, с помощью которого можно извлечь необходимые данные.

Для выполнения манипуляции над данными администратор систем-источников данных реализует процедуру, в теле которой выполняются нужные манипуляции. В качестве входных параметров передаются опции запроса.

Для извлечения небольших по объему данных, вместо передачи курсора в процедуру, можно использовать хранимые функции, реализующие те же запросы, но возвращающие необходимые значения. Таким способом можно извлекать значения определенных полей, ограничивая доступ к остальным полям таблицы. Также функции могут возвращать строку, содержащую ошибку, при её наличии, и пустую строку или метку удачного завершения запроса, в зависимости от предпочтений разработчиков.

Выводы. Обращение с базой описанным способом обеспечивает следующие преимущества:

- унификация способов обращения разработчиков к данным, предоставление интерфейса, разработчикам не обязательно знать о строении базы данных, так как логика загрузки данных реализуется администраторами данных;
- первичное предотвращение SQL-инъекций, так как в процедуру лишь подается параметр запроса, а не его логическая часть, то есть не используется конкатенация строк;
- детальное ограничение доступа к определенным данным и манипуляции с ними;

Пример реализации. Реализовать данную концепцию можно с использованием любого процедурного языка. В описанном ниже примере используется процедурный язык *Oracle Database – PL/SQL*, часть серверного приложения реализована с использованием языка *Java 1.7* и его библиотек *java.sql* и *oracle.jdbc*.

В контексте примера имеется пользователь *dbadmin* с неограниченными правами, в схеме которого имеется таблица *USERS* с полями *ID*, *NAME*, *BIRTHDAY*. Необходимо предоставить пользователю *appserver*, соответствую-

щему разработчикам серверного приложения, имеющему права только на создание сессии, возможность работы с данными из этой таблицы.

На Рисунке 2 от имени *dbadmin* создаются две хранимые процедуры. Процедура *GET_USERS* открывает переданный ей курсор. Процедура *UPDATE_USER* изменяет поле *NAME* по идентификатору записи.

```
--Создание хранимых процедур.
CREATE OR REPLACE PROCEDURE
  GET_USERS (CUR_REF OUT SYS_REFCURSOR, NAME_LIKE IN VARCHAR2) AS
BEGIN
  IF NOT CUR_REF%ISOPEN THEN
    OPEN CUR_REF FOR SELECT "ID","NAME","BIRTHDAY" from "DBADMIN"."USERS";
  END IF;
END;

CREATE OR REPLACE PROCEDURE
  UPDATE_USER (USER_ID IN NUMBER, NEW_NAME IN VARCHAR2) AS
BEGIN
  UPDATE "DBADMIN"."USERS" SET "NAME" = "NEW_NAME" WHERE "ID" = "USER_ID";
END;
```

Рис. 2 – Создание хранимых процедур

Далее, администратор базы данных создает публичные синонимы для созданных процедур и назначает права на их выполнение пользователю *appserver* (Рисунок 3).

```
--Создание публичных методов.
CREATE OR REPLACE PUBLIC SYNONYM "PS_GET_USERS"
  FOR "DBADMIN"."GET_USERS";
CREATE OR REPLACE PUBLIC SYNONYM  "PS_UPDATE_USER"
  FOR "DBADMIN"."UPDATE_USER";
--Назначение прав на выполнение процедур.
GRANT EXECUTE ON "PS_GET_USERS" TO "APPSERVER";
GRANT EXECUTE ON "PS_UPDATE_USER" TO "APPSERVER";
```

Рис. 3 – Создание синонимов и назначение прав на выполнение

Используя учетную запись *appserver*, разработчик серверного приложения создает подключение к базе данных, используя класс *Connection*, и хранит его в переменной *connection*. Чтобы получить данные из таблицы, необходимо открыть курсор при помощи процедуры *GET_USERS* и привести его к типу *ResultSet* (Рисунок 4).

```
public void getusers(String userLike) throws SQLException {
        String sql = "BEGIN PS_GET_USERS(?,?); END;";
        CallableStatement statement = connection.prepareCall(sql);
        statement.registerOutParameter(1, OracleTypes.CURSOR);
        statement.setString(2, userLike);
        statement.executeUpdate();
        ResultSet rs = (ResultSet) statement.getObject(1);
        while (rs.next()) {
            /*Логика работы с полученными данными.*/
        }
        /*Высвобождение ресурсов.*/
}
```

Рис.4 – Реализация обращения к процедуре *GET_USERS*

Таким же образом происходит обращение к процедуре *UPDATE_USER*, но без использования *ResultSet* (Рисунок 5).

```
public void updateUser(int userId, String newName) throws SQLException {
        String sql = "BEGIN PS_UPDATE_USER(?,?); END;";
        CallableStatement statement = connection.prepareCall(sql);
        statement.setInt(1, userId);
        statement.setString(2, newName);
        statement.executeUpdate();
        /*Высвобождение ресурсов.*/
}
```

Рис.5 – Реализация обращения к процедуре *UPDATE_USER*

Литература

1. Этапы разработки и внедрения информационно-аналитической системы [Электронный ресурс] – Режим доступа: http://www.prj-exp.ru/dwh/dwh_stages_of_development.php [Дата обращения 27.01.2017].

2. Джейсон, К. Oracle Certified Professional™ Подготовка администраторов баз данных [Текст] / Джейсон К., Ульрике Шв. – М.:Лори, 2009. - 868 с.

3. CREATE SYNONYM [Электронный ресурс] – Режим доступа: https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_7001.htm [Дата обращения 07.02.2017].

4. Хабибуллин, И. Ш. Java 7 [Текст] / И. Ш. Хабибуллин. – СПб.:БХВ-Петербург, 2012. – 768 с.

О.В. Прохорова

ПРОВЕРКА БОЛЬШИХ ЧИСЕЛ НА ПРОСТОТУ

(Самарский государственный технический университет)

Раскладывание числа на простые сомножители практикуется достаточно широко, например, в алгоритме RSA, где стойкость схемы RSA зависит от того, как быстро можно разложить большое составное число на 2 простых сомножителя. Процедура такого разложения занимает относительно много времени в силу большого числа итераций подбора, что является большим минусом в применении.

Предлагаемая автором методика проверки больших чисел на простоту основывается на применении сформулированных правил генерации простых чисел, 3 теорем и алгоритма. Представим правила сцепления цифр в числа:

1. Сцепляются два числа. Самая правая цифра есть только нечетное число, за исключением цифры 5, т.е. последней цифрой справа могут быть только: 1, 3, 7, 9. Слева это любые числа, начина с 0 и далее. Для двух разрядного числа и левой цифре 0 правая цифра 5 допускается.

2. Для каждого сцепленного числа $X = \{a_1 \cdot a_2 \cdot \ldots \cdot a_n\}$ находится отображение b = F(X) = F{a1 a2 a3 … an} , где ai – есть цифры числа X, b – есть