УДК 004.6

# СТРУКТУРА БАЗЫ ДАННЫХ В СИСТЕМЕ ТЕСТИРОВАНИЯ

А.А. Теут, студент,
Н.В. Силаев, доцент кафедры информатики и прикладной метематики,
Брестский государственный университет имени А.С. Пушкина

1. Предпосылки создания

Большинство профессиональных систем тестирования, реализованных за последние пять—семь лет (включая и ту, которая в настоящее время используется в нашем университете) используют для хранения информации о результатах тестирования базы данных с поддержкой запросов языка SQL, такие, как Microsoft SQL Server или mySQL.

Однако у большинства этих баз данных имелся один небольшой недостаток — они, были, как правило, изолированы от баз данных университета, где хранится информация о преподавателях и студентах. В результате это ставило перед администратором проблему конвертирования и дублирования данных. К тому же, даже после переноса данных из одной базы данных в другую оставалась актуальной проблема синхронизации. Ведь хотя базу данных с информацией о студентах университета и не приходится обновлять так часто, как, к примеру, базу данных со сведеньями о текущих грузоперевозках, однако после каждой сессии необходимо обновлять данные практически о каждом студенте — был ли он переведён на следующий курс, или ушёл в академический отпуск или был отчислен.

Поэтому было крайне желательно, чтобы в качестве базы данных для системы тестирования можно было бы использовать действующую базу данных студентов университета, добавив к ней несколько новых связей и таблиц. Одновременно следует учесть, что к той же базе данных будут иметь доступ другие приложения, и в ней будут содержаться много других таблиц и связей, которые не должны конфликтовать с исходными. Могут измениться названия таблиц, полей, сам SQL-сервер на котором установлена база данных.

Поэтому архитектура доступа к базе данных должна быть устроена таким образом, чтобы при внесении сколь угодно сложных изменений в базу данных изменения в самом приложении были по возможности минимальны.

2. Архитектура базы данных

Одной из особенностей базы данных среды тестирования является то, что в ней чаще изменяются не столько сами данные, сколько связи между ними. Фактически, в процессе семестра может идти обновление только данных о тестах и результатах тестов, а после сессии происходит глобальное перераспределение связей — большинство студентов перемещается на курс выше, а оставшиеся уходят в академический отпуск или отчисляются. Помимо этого, она должна быть открыта для возможных изменений вплоть до глобальной перестройки внутренней архитектуры. Поэтому прямой вызов запросов языка SQL из приложения был в нашем случае неприменим — ведь название полей, таблиц и связи между ними могут в любой момент поменяться.

Поэтому между таблицами и приложением был вставлен дополнительный слой из процедур, и отображений. Отображения получают информацию из таблиц, а процедуры записывают информацию в таблицы и получает информацию из отображений. Вся система позволяет перестраивать и исправлять базу данных, не выводя из строя само приложение, а также увеличивает степень абстракции, позволяя приложению использовать сервера баз данных в том числе и отличные от того, на котором его тестировали.

Однако сами тесты целесообразней хранить в отдельных файлах на жёстком диске, а в базе данных давать только ссылку на полный путь к ним, что позволит избежать засорения базы данных устаревшими данными, которые больше не используются, но всё равно остаются (как это часто случается при использовании систем автоматического создания статистики)..

Разумеется, аналогично могут быть устроены и другие приложения, использующие текущую базу данных, а набор файлов с командами языка SQL, которые обеспечивали бы создание всех дополнительных таблиц и процедур, можно будет включать напрямую в её дистрибутив.

Всё это позволит обеспечить совместимость новых версий, повысить совместимость приложения и облегчить переход не только на новые версии, но в том числе на новые разработанные системы тестирования. Разработчики новых программ смогут сосредоточиться на разработке кода и создавать у себя на жёстком диске всего лишь упрощённую версию текущей базы данных. Программы, разработанные под её процедуры, будут совместимы в том числе и с «большой», университетской базой данных.

3. Уровни архитектуры базы данных

Уровень классов – уровень, с которым непосредственно работает приложение. В случае смены стандарта базы данных, версии программного продукта, который обеспечивает работу сервера базы данных, платформы, на

которой она работает и тому подобных событий, будет достаточно внести исправления в классы этого уровня, не затрагивая основных классов приложения.

Уровень процедур – SQL –процедуры, которые получают в качестве параметров различные данные и заносят их в таблицу или извлекают из таблицы нужные данные и передают их на уровень вверх. Даже те из средств современной версии языка SQL, которые поддерживаются во всех, даже самых примитивных серверах баз данных, позволяют исполнять в том числе и сложные запросы.

Уровень отображений – необязательный уровень, который призван упростить составления процедур за счёт того, что между процедурами и таблицами вставляется уровень из объектов отображений (views), которые объединяют малые таблицы в единое целое.

Уровень таблиц – собственно, уровень данных, состоящий непосредственно из таблиц, как доступных из приложения напрямую, так и недоступных из него (к недоступным можно отнести, например, таблицы, сохраняющие статистику).