УДК 004.6

ОБЩАЯ АРХИТЕКТУРА КЛАССОВ ДОСТУПА К БАЗАМ ДАННЫХ

Теут А.А.
УО «Брестский государственный университет им. А.С.Пушкина», г. Брест
Научный руководитель – Силаев Н.В., доцент

В настоящее время абсолютное большинство программ, разрабатываемых программистами-прикладниками, ориентировано на работу с базами данных. Вместе с тем, программирование баз данных сопряжено с определенными особенностями, которые могут быть незнакомы разработчику, работавшему прежде лишь в области программирования математических задач или системного программирования.

Дело в том, что в математических задачах источником данных служит консоль и, что чрезвычайно редко, файл. Аналогичным образом при разработке системных приложений информация чаще всего извлекается либо из файлов, либо из системного реестра. Все эти источники данных объединяет одна особенность – за исключением тех случаев, когда файл расположен в сети, доступ к ним сравнительный быстрый и все команды доступа жёстко стандартизированы, т.к. являются функциями-членами стандартных класссов используемого языка программирования.

В отличие от описанных выше случаев, программирование баз данных имеет свои особенности, не всегда очевидные для человека, который только начинает с ними работать.

Во-первых, скорость доступа к информации, содержащейся в базе данных, всегда существенно ниже, чем к файловой информации.

Во-вторых, язык запросов SQL, на котором программируются операции доступа к базе данных, существенно отличается по своей структуре от обычного программного кода. Вне зависимости от языка, на котором ведётся разработка программы (это может быть C++, Java, PHP, C#), перед программистом встаёт задача чёткой и аккуратной реализации классов для работы со сложной базой данных, в которой содержится множество таблиц и связей между ними.

Нередко начинающий программист пишет код доступа к базе данных «напрямую», прямо в обработчике нажатия кнопки, загрузки страницы и т.п. Заметим, что такой способ сложен, не технологичен и соответствует современным требованиям стиля программирования. Дело в том, что при подобном подходе, код, вызывающий базу данных, оказывается «рассеянным» по всей прикладной программе, так как в этом случае информация из базы данных вызывается напрямую в те моменты, когда она бывает затребована алгоритмом.

Недостатки такого подхода в следующем: в случае, если в базу данных систематически вносятся изменения, а без подобных операций практические задачи немыслимы, программисту необходимо модифицировать команды SQL исходного кода программы. В связи с этим возникает очередная проблема: если даже исходный код программы размещён в едином файле, легко пропустить какой-либо из обращений к базе данных, даже если приложение сравнительно небольшое. Помимо этого, традиционно достаточно сложной является проблема тестирования работы всех операций вызова. Дело в том, что для большинства языков программирования, включая и языки высокого уровня, команда SQL является дополнительной переменной типа String. Поэтому практически все компиляторы (исключая некоторые расширения для C#, сделанные в Microsoft Visual Studio .Net 2008) не осуществляют даже синтаксической проверки SQL кода.

В связи со сказанным, для языков высокого уровня, включающих средства объектно-ориентированного программирования, разумно выделять классы, работающие с базой данных, в отдельный модуль. Для языков без поддержки наследования классов (таких, к примеру, как PHP), где доступ к базам данных реализован «напрямую» (то есть, HTML-таблица отображается сразу), самым удачным вариантом будет отдельный класс, в котором записаны все вызовы операций, разбитые по таблицам, для которых они применяются. Нами введен, в частности класс, содержащий, например, такие методы-функции, как добавление элемента, обновление данных элемента, получение Item по ID, получение всех Item, преобразование строки БД в элемент типа Item, преобразование строки БД в коллекцию элементов типа Item.

Что же касается таких языков, как C#, в которых для отображения содержимого базы данных уже имеются готовые компоненты, распознающие не только таблицы, но и связные списки, то для них очень удобно использовать целый набор из классов (назовем его DBManager) для доступа к базе данных. Методы подобного класса получают непосредственный доступ к базе данных и возвращают необходимые таблицы, как элементы типа DataTable. Прочие же классы (назовем их %Type%DBAccessor, где Type – название типа, который необходимо возвращать), получают команды непосредственно из классов, отвечающих за обработку и отображение данных. Они возвращают связные списки того типа, к которому принадлежат запрашиваемые данные.

Описанный подход нами реализован при построении системы тестирования общего назначения. Подобная система требует обработки большого объема разнородной информации (списки типов тестирования (текущее, экзаменационное, тематическое), списки разделов тестирования, списки тем тестирования, списки вопросов теста и соответствующих ответов-подсказок, списки групп тестируемых, списки результатов тестирования для каждого участника этого процесса), хранящейся в базе тестирования. Упоминаемая система тестирования, развивающая традиции прежних систем, разработанных и использующихся на математическом факультете БрГУ им. А.С. Пушкина, в настоящее время проходит комплексные испытания и внедрение в учебный процесс. По материалам наших исследований были сделаны доклады на ряде научных конференций от университетских до республиканских.

УДК 681.01(075.6)

ALLFUSION PROCESS MODELER 7 (BPWIN) КАК СРЕДСТВО ФУНКЦИОНАЛЬНОГО МОДЕЛИРОВАНИЯ БИЗНЕС-ПРОЦЕССОВ

Титова Т.И.

УО «Белорусский государственный экономический университет», г. Минск

AllFusion Process Modeler 7 (BPwin) – мощный инструмент моделирования, который используется для анализа, документирования и реорганизации сложных бизнес-процессов. Модель, созданная средствами BPwin, позволяет четко документировать различные аспекты деятельности - действия, которые необходимо предпринять, способы их осуществления, требующиеся для этого ресурсы и др. Таким образом, формируется целостная картина деятельности предприятия - от моделей организации работы в маленьких отделах до сложных иерархических структур. При разработке или закупке программного