Международный Соломонов университет

ДОКЛАД

SQL Server. Разработка приложений и программные решения.

студента V курса

факультета компьютерных наук

Лоншакова Андрея

1998В рамках темы доклада будут рассмотрены следующие вопросы:

1. Создание курсоров и их использование.

2. Доступ к базам данных SQL Server с помощью приложений клиента.

3. Переход от приложений Microsoft Office 97 к SQL Server.

**Создание курсоров и их использование**

Возможно наилучшее средство, добавленное корпорацией Microsoft в SQL Server 6.0, – это поддержка курсоров сервера. С помощью курсора можно обрабатывать отдельные строки в наборе, что не происходит при обычном применении команд SQL, которые работают для всех строк в наборе одновременно.

Разработчики Microsoft добавили полную реализацию курсоров сервера, которыми управляет база данных. Доступ к курсорам легко представляется с помощью инструментов разработки клиентских приложений и платформ, таких как SQLWindows или PowerBuilder. В предыдущих версиях SQL Server тоже можно было применять курсоры, но они поддерживались уровнем DBLibrary (сетевой библиотеки), а не сервером. По этой причине в документации Microsoft новые курсоры называются курсорами сервера (back-end cursors).

Курсор позволяет выполнять обработку результирующего набора внутри сервера без вмешательства клиентской программы для управления рабочим набором данных. Например, до появления SQL Server 6.0 было очень сложно написать высокопроизводительное приложение, выполняющее множество действий с набором данных, поскольку передача данных в клиентское приложение производилась по одной строке. Приложение клиента несло ответственность за инициирование дальнейшей обработки каждой строки. Курсор предоставляет возможность более усовершенствованного использования хранимых процедур Transact- SQL. Описанный выше алгоритм обработки строк реализуется без возвращения к клиенту.

Курсоры могут сильно замедлить производительность приложения. По своей природе они выполняются значительно медленнее, чем стандартные операции, основанные на наборе данных. Перед тем как приступить к разработке курсоров необходимо убедится, что не могут быть применены никакие другие операции с набором данных. Следует учесть не только легкость использования курсоров, но также и возможности уменьшения производительности вследствие их применения.

**Отличие между курсорами сервера и клиента**

В SQL Server доступны два типа курсоров: курсоры клиента (front-end or client cursors) и курсоры сервера (back-end or server cursors).Существующее между ними отличие имеет важное значение.

При написании приложения очень часто появляется необходимость выполнить определенную операцию на наборе данных. Для этого обычно применяется инструкция Update при изменении данных или инструкция Delete при удалении данных. Эти операции с наборами данных обычно обеспечивают высокую гибкость в приложениях, где предложение where может соответствующим образом определить требуемую задачу.

Предположим, что в базе данных Pubs вы хотите изменить почтовый индекс на 94024 для всех авторов, которые живут в Menlo Park. Для этого можно применить простую инструкцию Update, показанную ниже:

Update AUTORS

Set ZIP = '94024'

Where City = 'Menlo Park'

Go

C другой стороны, что если необходимо выполнить различные виды операций па наборе данных? Возможны два решения: можно выполнить множество операций на эксклюзивных наборах или можно получить полный набор данных и на основании величин в этом наборе выполнить требуемые операции. Второе решение является концепцией, лежащей в основе обработки с помощью курсора.

Обновлений и удалений наборов данных на сервере может быть недостаточно, так как обновления могут производиться в одной и той же строке несколько раз. В качестве альтернативы в базе данных может быть создан вид, называемый курсором.

Наибольшее преимущество обработки с помощью курсора состоит в возможности выполнить условные операции на определенной строке данных в наборе, независимо от других строк этого набора. Команды SQL выполняются на наборе данных, состоящем из одной строки. Сложные приложения часто требуют подобной детализации при обработке. Кроме того, эта детализация имеет несколько положительных моментов:

* Производительность. Операции, основанные на наборах данных тяготеют к применению больших серверных ресурсов, чем операции курсора.
* Лучший контроль транзакций. При обработке набора данных можно контролировать, что происходит с любой данной строкой независимо от других.
* Специальный синтаксис. Курсоры позволяют позиционированное обновление и удаление, которые применяются к строкам, выбранным в данный момент. Такие курсоры напрямую находят строки таблицы, поэтому исчезает необходимость индексирования.
* Эффективность. При выполнении нескольких операций на большом наборе данных, таком как вызов множества хранимых процедур, лучше, когда база данных обрабатывает данные и выполняет все действия с каждой строкой. Это более эффективно, чем выполнение каждой из задач последовательно по всему набору данных. Причины эффективности заключаются в кэшировании данных.

**Понятие курсора клиента**

Перед тем как был создан SQL Server 6.0, разработчики Microsoft пришли к выводу, что клиентам необходимо предоставить возможность обрабатывать данные и передвижение вперед и назад по результирующему набору. Пользователи нуждались в этих функциях для поддержки комплексных приложений, которые использовались при просмотре информации, выбранной из базы данных.

Корпорация Microsoft не могла включить в SQL Server более ранних версий курсоры сервера, которые поддерживают некоторые другие производители, поэтому было принято решение смоделировать некоторые из функций курсора в интерфейсе программирования клиентских приложений для базы данных SQL Server – DBLibrary.

Корпорация Microsoft унаследовала интерфейс клиента DBLibrary, позволяющий взаимодействие с базой данных SQL Server, от Sybase. DBLibrary – это набор команд и функций, которыми можно пользоваться в среде языка программирования С для выполнения операций с базой данных. В SQL Server 6.0 и в последующих версиях разработчики Microsoft изменили интерфейс DBLibrary к базе данных на интерфейс ODBC.

Для получения подобной функциональности разработчики Microsoft добавили курсоры к наборам данных на стороне клиента.

Эти курсоры работают с помощью DВLibrary позволяющей обычное взаимодействие с базами данных: считывание информации из потока данных таблицы (Tabular Data Stream, TDS) выполняется настолько быстро, насколько требует клиент. TDS – это метод связи применяемый DBLibrary для выборки информации из базы данных. Обычно DBLibrary передает всю информацию, которая была выбрана из базы данных , приложению клиента, которое, в свою очередь, выполняет дополнительную работу. С помощью курсоров DBLibrary кэширует эти записи сама, до того, как клиент не отменит просмотр данных с помощью курсора.

Подобное кэширование имеет несколько ограничений:

* SQL Server не обладает средством контроля или минимизирования блокировок, устанавливаемых в базе данных. Поэтому в курсоре блокировки устанавливаются на все страницы данных, а не только на те, где выполняется обновление. Это происходит потому, что SQL Server не может установить, какие еще операции, помимо инструкций select, выполняются в базе данных.
* При работе с большими наборами данных, кэширование может очень быстро поглотить ресурсы на стороне клиента.
* При обработке больших объемов данных кэширование неэффективно, так как все данные, участвующие в обработке, пересылаются по сети.

До момента, пока корпорация Microsoft не разработала курсоры сервера, это средство SQL Server оставалось чисто декларативным и не было предназначено для серьезного применения. Курсоры сервера обладают всеми преимуществами курсоров клиентов, но лишены характерных для них накладных расходов и ограничений. Помимо соображений обратной совместимости, других веских причин применения курсоров клиентов в приложениях SQL Server 6.5 не существует.

Применение курсоров сервера включает в себя пять состояний, описанных ниже:

|  |  |
| --- | --- |
| Состояние | Объяснение |
| DECLARE | На этом этапе SQL Server проверяет, верный курсор выберет нужные для запроса данные. SQL Server создает в памяти разделяемую структуру, содержащую определение курсора, необходимое для компиляции на этапе open. |
| OPEN | SQL Server начинает отвечать на выполнение инструкции declare. Он распознает запрос и заносит идентификаторы строк во временную рабочую область, которая будет использоваться клиентом при считывании идентифицируемых курсором строк. |
| FETCH | В этом состоянии курсор возвращает данные, необходимые для выполнения любой требуемой операции. |
| CLOSE | SQL Server закрывает открытый ранее курсор и освобождает любые блокировки, которые могли быть установлены в результате его открытия. |
| DEALLOCATE | SQL Server освобождает разделяемую память, используемую инструкцией declare, не позволяя больше другим процессам выполнять инструкцию open. |

Для работы с курсорами SQL Server необходимо следовать состояниям, описанным выше. Сначала необходимо объявить (declare) курсор. Затем вы можете его открыть (open) и считать данные (fetch). При считывании данных (состояние fetch) можно выполнить с активной строкой несколько операций. Завершив работу с курсором, вы должны закрыть (close) его и освободить память (deallocate). Это делается для того, чтобы SQL Server не расходовал ресурсы на управление ненужным курсором.

**Практическое тестирование**

На первый взгляд курсоры кажутся мощным средством программирования приложений. Конечно, они предоставляют приложениям развитую функциональность, предназначенную для просмотра данных. при этом устраняется необходимость в программировании подобных функций на стороне клиента.

Применение функций курсоров сильно сокращает время разработки приложений, но при этом сильно падает производительность системы. хотя реально производимый эффект в значительной степени зависит от конкретной реализации, включай и аппаратуру, на которой работает сервер, курсоры могут очень сильно уменьшить скорость работы приложения.

**Доступ к базам данных SQL Server с помощью приложений клиента**

В SQL Server 6 корпорация Microsoft начала поставку нового интерфейса с клиентом, работающего с помощью ODBC. Вместо использования старого разработанного Sybase подхода, предполагающего применение DBLibrary, теперь появился новый путь доступа – через ODBC.

Так как корпорация Microsoft является владельцем спецификации ODBC, для нее не составило большого труда переработать и оптимизировать продукт для доступа в SQL Server. Новый интерфейс полностью лишен проблем, связанных с падением производительности, всегда ассоциировавшейся с ODBC. Корпорация Microsoft все еще поддерживает интерфейс DBLibrary, но только для обратной совместимости. Но в настоящее время интерфейс ODBC становится стандартом для доступа к базам данных.

**Конфигурирование ODBC для SQL Server**

Интерфейс ODBC – это способ соединения различных сервисов данных с различными приложениями клиентов универсальным образом. С момента его появления на рынке в 1991 году, ODBC подвергся значительным изменениям.

Конфигурирование интерфейса ODBC выполняется с помощью апплета в Control Panel: ODBC или ODBC32. Процесс установки конфигурации ODBC для Windows NT Workstation или Server версии 4.х (или выше) и для Windows 95 идентичен. В Windows NT Workstation или Server версии 3.51 (или меньше) информация диалогового окна остается той же самой, хотя могут быть отличия в выделении значений м расположении опций close/minimize/restore.

Для конфигурирования ODBC при работе с Microsoft SQL Server 6.5 выполните следующие действия:

1. Установите драйверы ODBC, которые поставляются с SQL Server 6.5.
2. Из меню Start выберите Settings, Control Panel.
3. Выберите мышью пиктограмму ODBC. Появится конфигурационное диалоговое окно.
4. Новые источники данных, доступ к которым предоставляется с помощью ODBC, добавляются в диалоговом окне Add Data Source. В диалоговом окне Data Source нажмите кнопку Add.
5. Выберите из списка SQL Server и нажмите кнопку OK. Появляется диалоговое окно ODBC SQL Server Setup.
6. Введите имя источника данных, например, LocalServer. Имя может быть любым сочетанием разрешенных символов, которое несет определенную информацию об источнике данных.
7. Введите описание типа данных ODBC – MS SQL Server 6.5. Это необходимо для определения, какой источник данных предоставляет этот сервис ODBC без использования имени этого источника.
8. Введите имя SQL Server, где находятся данные. Если SQL Server работает локально на Windows NT, можно ввести (local). Драйвер ODBC найдет сервер с помощью протокола именованных каналов.
9. Введите сетевой адрес или сетевую библиотеку, если администратор сети/базы данных указывает, что это необходимо сделать. Обычно эта установка может быть оставлена со значением "Default" и драйвер ODBC найдет сервер при первом присоединении.
10. Введите имя базы данных, к которой должен присоединится сервис ODBC. Некоторые программы – клиенты ODBC – не обладают возможностью изменять рабочую базу данных с помощью команд ODBC. Поэтому может возникнуть необходимость определить источник данных для каждой рабочей базы данных, установленной на одном и том же сервере.
11. Установка языка и кодовой страницы должны быть оставлены без изменений, если не существует веской причины для их изменения.
12. Для того, чтобы добавить новый источник данных, нажмите кнопку OK. К списку доступных источников данных будет добавлен новый сервер.

**Использование продукта Delphi 2 производства компании Borland**

Продукт Delphi 2 производства компании Borland – это мощный инструмент разработки приложений клиента в архитектуре клиент/сервер, который помогает облегчить работу с источниками данных ODBC. Продукт Delphi 2 основан на Pascal и является самой последней версией, которая была выпущена в 1996 году (на данный момент существует версия 3 этого продукта). Программная система Delphi 2 является средой разработки, поддерживающей 32-разрядную архитектуру. Его высоко ценят за легкость в работе и интеграции.

**Установление соединения**

Продукт Delphi 2 имеет собственные драйверы для SQL Server 6.5,работающие с Microsoft ODBC. Самый простой способ подготовки Delphi 2 для работы с базой данных состоит в применении апплета Database Explorer, поставляемого вместе с Delphi 2.

Для подготовки интерфейса ODBC с помощью Database Explorer активизируйте Delphi и выберите Database, Explorer.

Из меню Object выберите New. Выберите конфигурацию SQL Server.

Выберите мышью редактируемую часть свойства DATABASE NAME в правой панели и введите имя рабочей базы данных.

В свойстве USER NAME введите имя пользователя, который будет присоединятся к этой базе данных по умолчанию. Кроме того, для списка свойств диалогового окна обновите установки драйвера ODBC.

Из меню Object выберите Apply. Активизируется рабочая база данных. Вы заметите, что стрелка, указывающая неактивную рабочую базу данных, исчезнет.

Проверьте, что все установлено правильно. Затем выберите мышью знак плюс, находящийся слева от имени псевдонима базы данных. Откроется вид Explorer. Здесь будет отображено окно Database Login.

Введите пароль системного администратора. Раскроется дерево Explorer. Оно содержит различные компоненты SQL Server, доступные для манипуляции с помощью SQL Server.

**Понятие о модели базы данных Delphi 2**

Для построения интерфейса с базами данных Delphi 2 применяет уровневый подход. Он предполагает наличие нескольких уровней абстракции от базы данных, что позволяет сформировать общий интерфейс для прикладного программирования, состоящий из общедоступных и знакомых объектов.

Продукт Delphi 2 содержит определенное число классов, применяемое для работы с данными. Эти классы отвечают за выполнение запросов, осуществляющих необходимое манипулирование информацией. Классы, которые могут быть применены для этих целей варьируются от TTable, представляющего таблицу, до TQuery, активизирующей индивидуальный запрос, связанный с элементами управления, составляющими интерфейс пользователя.

Классы интерфейса с данными обычно в форме не видны. Физический источник данных обычно связывается с определенным классом, который отвечает за взаимодействие с объектами/элементами управления интерфейса пользователя, такими как поля и списки. Это взаимодействие осуществляется через не визуальный класс TDataSource, который передает данные из класса физических данных в визуальные объекты и элементы управления, связанные с данными (data-bound controls), которые помещены в форму редактирования.

Манипулирование данными с помощью классов TQuery или TTable состоит в выборке записей, их обновления и так далее. Подобные операции могут быть выполнены с помощью методов, которые имеют данные классы или путем добавления в форму редактирования элемента управления типа Navigation. TDBNavigator – это класс, который взаимодействует с TDataSource и содержит стандартные кнопки манипуляции данными в форме: Next, Previous, Insert и Update. Поместив один из этих элементов управления в форму редактирования и связав его (на этапе разработки, установив его свойство или на этапе выполнения, переустановив свойство на требуемый в форме источник данных), вы получите все необходимые компоненты для построения окна редактирования структуры данных. Эта структура может быть либо запросом. либо таблицей базы данных.

Наконец. в Delphi 2 существует необходимость в формировании элементов управления, предназначенных для просмотра или редактирования данных. Delphi 2 содержит все стандартные элементы управления для редактирования, включая списки и окна со списком, которые могут быть связаны с определенным TDataSource. Необходимо только поместить элемент управления в форму и установить свойства DataSource и DataField.

**Использование Microsoft Visual Basic 4 и 5 версий**

Visual Basic 5(VB5) – это самая последняя версия стандарта BASIC корпорации Microsoft. С появлением VB5 началось широкомасштабное включение VB в другие среды разработки приложений, такие как PowerPoint, Excel и Word. Кроме того, VB5 обеспечивает связи приложений и программных инструментов с Internet.

Как инструмент VB5 имеет большой набор функций и огромное количество установок. Кроме того, этот продукт включает несколько мастеров по генерации форм, позволяющих создавать некоторые общие формы, применяемые в качестве базы разработки.

VB5 также поддерживает работу с объектными библиотеками, предназначенными для доступа в SQL Server. Это очень важные возможности, обеспечивающие мощную базу для разработки приложений клиент/сервер.

**Понятие о модели базы данных Visual Basic**

Visual Basic и Delphi 2 имеют похожие методы доступа к данным. Доступ контролируется объектом данных, находящимся в форме. В отличие от Delphi, VB не применяет истинно не визуальных классов объектов или контейнерных классов. Вместо этого имеется средство, позволяющее изменить видимое свойство и скрыть его в случае необходимости.

В VB с базами данных можно работать несколькими различными путями. Во-первых, если необходимо получить набор полей. связанных с базой данных и соединение, с помощью которого выполняется автоматическое обновление этих полей, можно применить элемент управления, связанный с данными. Подобный элемент управления позволит указать свойства соединения и затем связать элементы управления формы с этим элементом управления и ассоциированными с ним столбцами.

Другие альтернативы работы с базой данных в VB включают использование объектов и их ассоциированных свойств, коллекций и так далее.

**Переход от приложений Microsoft Office 97 к SQL Server**

Набор программ Microsoft Office 97 предоставляет средства разработки приложений, основанных на входящих в него инструментах. Теперь в Office 97 можно работать с Visual Basic в любом из программных продуктов набора – от Word до Excel и Access. В каждой из этих сред с помощью VB вы имеете возможность разрабатывать полноценные приложения.

Дальше речь пойдет о взаимодействии с SQL Server, а не управлении самим сервером. Доступ к SQL Server может быть в большей степени стандартизован с помощью объектов доступа к данным (Data Access Object). При работе с Access следует принимать во внимание специальные соображения. Короче говоря, возможность работы с данными в SQL Server всегда будет хорошим дополнением к вашему приложению. Кроме того, применяя для работы с информацией технологию клиент/сервер, можно увеличить мощность обработки данных, предоставив вашему приложению доступ к SQL Server.

**Переход от Access к SQL Server**

Инструмент перехода поможет перенести схему базы данных в SQL Server, переместив таблицы и отношения в объекты, основанные на SQL Server. Утилиты перенесут таблицы на сервер, осмотрев их и создав необходимые индексы и другие объекты, поддерживаемые в системе. Кроме того, могут быть созданы триггеры для сохранения правил целостности отношений, которые применяются в Access.

В результате в Access образуется система, содержащая связанные таблицы, которые ссылаются на таблицы в SQL Server.

Мастер не создает хранимых процедур, осуществляющих те же операции, которые выполнялись при манипулировании данными в коде Access Basic. В этом случае необходимо ознакомиться с дополнительной информацией, касающейся хранимых процедур и сравнить те средства, которыми они обладают с операциями. выполняемыми в коде Basic, где это возможно, старайтесь перенести инструкции манипуляции данными в хранимые процедуры.

**Использование Microsoft Access c SQL Server**

Продукт Microsoft Access дает возможность применять при разработке большое количество доступных для тестирования и реализации средств SQL Server. Так как Access обладает более слабым ядром и не является системой клиент/сервер, его применение в качестве системы сервера в больших проектах весьма ограничено. Но Access все же представляет собой мощную среду разработки при работе с базой данных.

Access содержит хороший интерфейс пользователя. с помощью которого можно разрабатывать таблицы и устанавливать отношения между ними. Существуют различия в том, как Access работает с базами данных, включая и аспекты физического хранения данных. При некоторых вопросов может отличаться терминология. Ниже приведены некоторые примеры подобных отличий:

* В Access не существует понятия устройства базы данных, как в SQL Server. Грубым подобием устройства базы данных SQL Server может служить устройство жесткого диска, на котором создана база данных. Базы данных в Access – это отдельные файлы, обычно имеющие расширение .MDB.
* В Access виды называются запросами. Так же, как и в SQL Server вы можете создать обновляемые запросы, выбирать строки с помощью запросов и создавать многотабличные объединения, на базе которых строить комбинирующие запросы.
* В SQL Server индивидуальный столбец представляет определенный элемент данных внутри строки. В Access строками называют наборы элементов данных, а столбцы называют полями.
* Типы данных в Access и SQL Server во многих случаях имеют различное название и различные характеристики.

Существуют и другие различия, но приведенные выше являются основными. Их знание помогут при использовании SQL Server и Access. Если приложение работает в Access, оно представлено шестью различными элементами. Для того, чтобы получить понятие о различных компонентах Access и принять решение об их переносе в систему SQL Server, рассмотрим следующую таблицу:

|  |  |
| --- | --- |
| Элемент | Описание и применение |
| Таблицы | То же, что и в SQL Server. Это возможно наиболее важный элемент, который будет перенесен. Таблицы содержат строки информации, которую вы хотите конвертировать в SQL Server. |
| Запросы | Запросы должны быть перенесены и конвертированы, но это будет в основном задача, выполняемая вручную. Это значит, что необходимо просмотреть код, лежащий в основе каждого запроса и создать в SQL Server соответствующий вид. |
| Формы | Формы не конвертируются в SQL Server. В модели клиент/сервер приложение клиента отвечает за управление интерфейсом пользователя. Сервер отвечает за управление обработкой данных. |
| Отчеты | Так же, как и формы, отчеты не будут конвертироваться в SQL Server. Исключение составляет отчет, основанный на запросе. Access позволяет в отчет включать запрос. Для получения наилучших результатов следует пересмотреть свойства отчета, конвертировать встроенный запрос и сформировать из него вид в SQL Server. Затем в запросе обращаться к новому виду. Производительность увеличится, а отчет будет работать так же. как он работал со встроенным запросом. Но в результате вы получите реализацию отчета в системе клиент/сервер. |
| Макросы | Макросы – это элемент жестко связанный с Access. Макросы могут вызывать запросы и выполнять некоторые высокоуровневые команды, которые должны быть пересмотрены. Так же, как и в ситуации с запросами, которые могут быть перемещены в SQL Server для увеличения производительности. |
| Модули | Решение о переносе модулей редко принимается автоматически. Это значит, что каждая процедура и подпрограмма переносится отдельно, после тщательной оценки необходимости этого шага. При переносе модулей вы располагаете несколькими вариантами. Подпрограмму можно конвертировать в хранимую процедуру, вид или динамический запрос. Это полностью зависит от той обработки, которая производится в базе данных. Помните о необходимости внесения изменений в короткие пакеты команд и продолжении тестирования. Перед переходом к следующему этапу конвертирования очень важно тщательно протестировать те функции системы, с которыми вы начали работать. |

**Создание приложений клиент/сервер с помощью Microsoft Access**

Microsoft Access не является системой клиент/сервер. Его приложения не содержат ярко выраженных компонентов клиента и сервера. Однако Access может быть использован в качестве инструмента разработки, предоставляя средство формирования интерфейса пользователя. Этот инструмент содержит такие мощные средства, как генератор отчетов, построитель запросов и так далее. Подобные инструменты очень полезны при разработке систем.

При создании приложения рассмотрите возможность первоначальной разработки его с помощью Access для базы данных, которую можно тестировать, разрушать и восстанавливать. Применяя подобный подход, Access может быть использован там, где он лучше всего подходит, оставляя управление базой данных для SQL Server.

Таблицы базы данных Access могут быть конвертированы и перенесены на SQL Server. Таблицы, запросы и виды – это объекты, которыми лучше всего управляет SQL Server. После создания доступных в Access объектов, они могут быть легко конвертированы в SQL Server.

Access становится инструментом разработки, предоставляющим средство формирования интерфейса пользователя и позволяющим работать с базами данных SQL Server.

**Создание транзитных запросов**

Запросы, созданные в Access и затем выполняемые на SQL Server, сначала проходят уровень соединения ODBC, а затем передаются SQL Server в виде временных хранимых процедур. Результаты выполнения возвращаются приложению и с ними уже можно работать.

Что случится, если вы захотите получить больше средств контроля над выходными данными? Что, если вы захотите вызывать хранимые процедуры и непосредственно работать с результатами? Подобные операции выполняются с помощью транзитных запросов (pass-through queries). Применяя Access, можно создать транзитные запросы, которые минуют трансляцию и прямо передаются в SQL Server. Для того, чтобы установить запрос как транзитный, сначала выделите его и выберите мышью Design. Затем из появившихся меню выберите Query, SQL Specific и наконец Pass-through. После этого появится окно, содержащее инструкцию SQL.

Транзитные запросы позволяют вызывать хранимые процедуры, обращаться к видам и применять другие средства SQL Server, которые могут быть недоступны при выполнении обычного запроса Access. Для выполнения хранимой процедуры, например, можно просто использовать команду EXECUTE.

Например, для вызова хранимой процедуры AccessProc из транзитного запроса, можно использовать следующую инструкцию:

exec accessproc

В результате хранимая процедура будет выполнена и результаты будут возвращены из запроса. Такой тип логики очень легко включается в приложение с помощью создания необходимых запросов или с помощью формирования запросов посредством Access Basic

**Практическое тестирование**

СУБД Access – это мощный инструмент. Он предоставляет возможность разработчикам корпоративных систем создавать надежные приложения баз данных, которые могут быть легко установлены в рабочую среду. Подобные приложения могут включать систему безопасности, иметь высокую производительность и дружественный пользователю интерфейс. С помощью Access процесс знакомства пользователя с приложением становится естественным и не вызывает затруднений.

После выполнения переноса баз данных Access в SQL Server, хорошее приложение становится еще лучше. Увеличивается производительность, улучшается система безопасности и доступность информации вашего приложения в масштабе корпорации.

В практических разработках использование Access в качестве инструмента разработки и тестирования дает хорошие результаты. Ниже приведен ряд соображений, которые нельзя игнорировать при тестировании систем баз данных, работающих с помощью Access:

* *Производительность в системах с мощной индексацией и большим количеством вводимой информации.* При переходе к SQL Server этот вопрос обязательно должен быть пересмотрен, так как система SQL Server предоставляет значительно большие возможности по оптимизации системы. Вы можете применить лучшие индексы и другие процессы стороны сервера, позволяющие увеличить производительность SQL Server. Подобные опции недоступны на том же уровне Access. Вы можете добавить индексы в Access, но это приведет к совершенно другим результатам.
* *Тонкая настойка запросов, видов и хранимых процедур*. Проведя масштабную работу по оптимизации хранимых процедур и других объектов, перенесенных из Access, можно значительно увеличить производительность системы по сравнению с тем, как система работала сразу после конвертирования. После завершения конвертирования и тестирования, просмотрите код хранимых процедур и видов. Убедитесь. что он предусматривает максимально широкое применение возможностей, предоставляемых SQL Server, служащих для повышения производительности и эффективности работы.
* *Безопасность*. Не забывайте, что при переносе приложения, вы должны управлять системой безопасности SQL Server.

После завершения конвертирования приложения необходимо пересмотреть ряд его элементов, касающихся стиля разработки. Лучше всего составить список этих элементов и проверять их в каждом случае перехода на SQL Server.