Михаил Фленов

bhv®

Transact-SQL

- Подробное описание языка
- Большое количество примеров
- Готовые решения типичных задач
- SQL для обслуживания 1С:Предприятия

Наиболее полное руководство

+

В ПОДЛИННИКЕ®

Михаил Фленов

Transact-SQL

Санкт-Петербург

«БХВ-Петербург»

2006

УДК 681.3.068
ББК 32.973.26-018.1
Ф69

Фленов М. Е.

Ф69 Transact-SQL. — СПб.: БХВ-Петербург, 2006. — 576 с.: ил.

ISBN 5-94157-790-7

Подробно рассмотрено использование языка Transact-SQL для администрирования и манипуляции данными СУБД Microsoft SQL Server. Материал сопровождается большим количеством практических примеров, написанных автором. Уделено внимание вопросам применения Transact-SQL при совместном использовании 1С и Microsoft SQL Server. На прилагаемом к книге компакт-диске размещены примеры запросов, тестовая база данных, а также дополнительная документация и статьи автора, посвященные базам данных.

Для программистов и администраторов СУБД

УДК 681.3.068
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Ирина Иноземцева*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Наталия Першакова*
Дизайн серии	*Игоря Цырульникова*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 21.02.06.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 46,44.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-790-7

Оглавление

Предисловие

Уже долгое время язык запросов SQL (Structured Query Language, структурированный язык запросов) является стандартом доступа к базам данных. Не имеет значения, какой язык программирования вы используете, я больше чем уверен, что доступ к данным на сервере баз данных происходит с помощью запросов SQL. Исключением могут быть только локальные таблицы типа DBF или Paradox. В них к данным можно обращаться благодаря драйверу через прямой доступ. Но и в этом случае драйвер способен поддерживать запросы, значительно расширяющие возможности работы с данными.

При работе с клиент-серверными или *n*-уровневыми системами доступ обязательно происходит именно через SQL-запросы. Более удобного и мощного средства пока не придумали. Даже там, где доступ, как вам кажется, идет напрямую, используется SQL, просто среда разработки прячет от нас запросы.

Хорошее знание баз данных и умение писать эффективные запросы позволит вам создавать действительно быстрые и полезные приложения. Помимо этого, программистам очень часто приходится выполнять какие-либо одноразовые задания, и SQL позволяет сделать все быстро и качественно.

Язык запросов был стандартизирован еще в 1992 году. За это время его возможности немного устарели, но не потеряли своей актуальности. В конце 90-х годов предпринимались попытки принять обновленный стандарт, но война между различными производителями баз данных не позволила достичь компромисса. В связи с этим язык SQL получил два вида расширений: Transact-SQL или T-SQL (поддерживается Microsoft) и PL/SQL (яркий представитель — Oracle). Каждый из этих производителей максимально придерживается стандарта SQL 1992 года, и все запросы на этом языке будут выполняться корректно. Но для предоставления пользователю новых возможностей добавлены новые команды, которые объединены под именами Transact-SQL и PL/SQL и поддерживаются на разных базах данных.

Рассмотреть абсолютно все команды и возможности этих стандартов нереально. Поэтому мы ограничимся стандартом 1992 года и расширением Transact-SQL, т. к. серверы, созданные компанией Microsoft, получили в нашей стране достаточно широкое распространение и продолжают завоевывать сердца разработчиков. Рассматривать всю спецификацию SQL тоже не имеет смысла, потому что большая ее часть относится к разработчикам серверов баз данных (какими должны быть поля, их типы, размерность и т. д.). Мы же будем рассматривать стандарт с точки зрения программистов конечных приложений, которые уже используют язык SQL, а не реализуют его в своих программах.

Если у вас возникли вопросы или пожелания по книге или языку SQL, жду ваших писем и отзывов по адресу **horrific@vr-online.ru**. Все мои последние работы основываются на вопросах и предложениях читателей, с которыми я регулярно общаюсь на форуме сайта **www.vr-online.ru**. Если у вас появятся какие-то вопросы, то милости прошу на этот форум. Я постараюсь ответить по мере возможности и жду любых комментариев по поводу этой книги. Ваши замечания позволят мне сделать эту и последующие работы лучше.

Благодарности

Работа над книгой — достаточно тяжелый труд, который отнимает очень много сил. Каждую свою работу я стремлюсь сделать лучше предыдущей, а это уже затраты не только сил, но и времени. Если бы не моя семья, то выход этой книги задержался бы, наверное, на месяц. Лето 2005 года, когда писалась эта книга, выдалось для меня достаточно тяжелым, потому что за три месяца к нам по очереди приезжали гости и родственники из разных городов. В это время дом превращался в сумасшедший, и работать приходилось по ночам. Если бы не помощь семьи, то книга точно была бы задержана.

Перед сдачей книги из-за отсутствия времени я иногда срывался на своих родных (жену, родителей). Сейчас, оглядываясь на прошлое, хочется попросить прощения за мои срывы.

Пять лет назад я изучал MS SQL Server в МГТУ им. Баумана в Москве, где курс читал Гилев Алексей Вячеславович. Хочется поблагодарить его за то, что дал мне основные знания программирования на языке Transact-SQL. До того момента я работал только с классическим ANSI SQL, а этот курс помог мне расширить знания.

Отдельное спасибо редакции "БХВ-Петербург" за то, что помогают в издании моих работ. Редакторам и корректорам за то, что выискивают мои "баги" и делают мой технический русский немного более литературным и читаемым.

Как всегда, благодарю своих друзей из редакции журнала "Геймлэнд", всех друзей по сайту **www.vr-online.ru**. С каждой новой книгой и с каждым годом количество друзей растет, и чтобы поблагодарить всех, понадобится целая книга, а ведь каждый помогает мне своими знаниями, поддержкой и просто советом. Поэтому, если я кого-то упустил, то хочу попросить прощения.

Для кого эта книга

Вполне логичный вопрос — кому будет полезна эта книга? Конечно же, администраторам и программистам. С программистами все ясно, они должны знать язык, с помощью которого можно получать данные от сервера. Но зачем это нужно администратору?

Начинающие администраторы для управления сервером SQL очень часто используют специальную утилиту Enterprise Manager, которая предоставляет визуальный интерфейс и удобство в администрировании сервером. Визуальность — это хорошо, но сценарии лучше. Я сам в этом убедился, когда нужно было тиражировать схожие настройки базы данных на несколько серверов. Сначала я копировал базу с помощью резервного копирования и восстановления на новый сервер, а затем чистил новую базу, убирая ненужные данные. Это долгий и не очень удобный процесс.

Чтобы ускорить тиражирование, я написал один сценарий, который последовательно выполнял все необходимые действия — создание базы данных, процедур, функций и индексов. Этот сценарий выполняется намного быстрее, потому что не надо копировать избыточные данные и чистить таблицы. После этого я сохраняю на диске все сценарии создания базы данных и изменения ее настроек. Это позволяет быстро создать новую базу данных.

Язык SQL необходимо знать и для тестирования производительности сервера. Оптимизация работы сервера входит в обязанности администратора, а значит, он должен уметь выполнять запросы, анализировать скорость их работы и повышать их производительность. Конечно же, оптимизировать код сценария должен программист, но скорость можно повысить и с помощью оптимизации базы данных, и это должен делать администратор. Например, если администратор увидит с помощью программы мониторинга сервера, что какой-то запрос выполняется достаточно часто, то он должен проанализировать его текст и выяснить, какие поля чаще всего используются для сравнения и, если необходимо, добавить соответствующие индексы. Это может в несколько раз увеличить производительность.

Введение в SQL

Сначала я хотел выделить отдельный раздел для вводной информации о языке запросов, но потом решил сократить эту информацию до минимума, а максимум внимания уделить практике.

С помощью SQL-запросов можно создавать реляционные базы данных. Этот язык стал стандартом, поэтому если вы хотите работать с базами данных, то должны знать его как свои пять пальцев.

Язык SQL определяется Американским национальным институтом стандартов (American National Standards Institute, ANSI) и Международной организацией по стандартизации (International Organization for Standardization, ISO). Но некоторые производители баз данных вносят изменения и дополнения в этот язык. Изменения незначительны, и основа остается совместимой со стандартом, а вот дополнений в разных базах данных может быть очень много. В Transact-SQL, который мы будем рассматривать в данной книге, очень много возможностей, которые не описаны стандартом SQL-92.

Что такое реляционная база данных? Это таблица, в которой в качестве столбцов выступают поля данных (поля определяют класс значений в соответствующей колонке таблицы), а каждая строка хранит данные. Строки очень часто в технической литературе называют записями. Здесь действует эффект разных языков. В английском языке строку таблицы в базе данных принято называть словом *record* (запись), потому что слово *line* (строка) имеет немного другой, более узкий смысл.

Давайте рассмотрим реляционную базу данных на примере табл. П1. Заголовок таблицы — это имена полей (колонок). Каждая строка — запись с данными. Реляционная база данных состоит из таких таблиц. На первый взгляд, все слишком просто, но эта простота позволяет решить практически любую задачу. Я, по крайней мере, еще не встречался с ситуацией, в которой реляционная база была бы бессильна.

Таблица П1. *Пример таблицы реляционной базы данных*

Фамилия	Имя	Отчество	Пол
Иванов	Иван	Иванович	М
Петрова	Мария	Анатольевна	Ж
Сидоров	Сергей	Петрович	М
...			

В каждой таблице должно быть одно уникальное поле, которое однозначно будет идентифицировать строку. Это поле называется ключевым. Эти поля очень часто используются для связывания таблиц. Но даже если у вас таблица не связана, ключевое поле все равно обязательно. Допустим, что вам необходимо изменить имя записи, где в поле `Фамилия` находится значение `Иванов`. Для этого нужно написать запрос, который будет реализовывать следующую логику: "Изменить поле `Имя` на значение `XXXX`, где поле `Фамилия` содержит значение `Иванов`". Именно такую логику будет использовать база данных. Но что произойдет, если в таблице есть две записи, в которых поле `Фамилия` содержит одно и то же значение? В этом случае будут обновлены все строки, в которых в поле `Фамилия` содержится `Иванов`.

Но обновление всех записей не всегда является необходимым. Чтобы однозначно идентифицировать строку, лучше использовать ключевые поля. База данных сама следит, чтобы данные в ключевом поле были уникальными для каждой строки. Для упрощения создания ключевых полей в большинстве баз данных есть специальный тип данных (например, счетчик), который может автоматически увеличивать значение для каждой новой строки, чем и достигается уникальность. В табл. П2 приводится пример таблицы базы данных, где первое поле — это автоматически увеличиваемое число.

Таблица П2. *Пример таблицы реляционной базы данных с ключевым полем*

Ключ	Фамилия	Имя	Отчество	Пол
1	Иванов	Иван	Иванович	М
2	Петрова	Мария	Анатольевна	Ж
3	Сидоров	Сергей	Петрович	М
4	Иванов	Иван	Иванович	М
...				

В табл. П2 я добавил еще одну запись, в которой все поля идентичны строке с номером 1. Это реальная ситуация, ведь вполне вероятно наличие двух человек с одними и теми же фамилией, именем и отчеством, но разной датой рождения. Теперь, чтобы обновить запись первого Иванова, необходимо написать запрос со следующей логикой: "Изменить поле `Имя` на значение `XXXX`, где ключевое поле содержит значение 1". Благодаря уникальному ключевому полю мы четко определяем запись, которую необходимо обновить.

Имена полей в базе данных также должны быть уникальными, но в этом случае не обязательно числовыми. Их можно называть как угодно, лишь бы уни-

кальность соблюдалась в пределах таблицы, а имя отражало содержащиеся данные.

Язык запросов SQL может быть двух типов: интерактивный и вложенный. Интерактивный вариант — это отдельный язык, он сам выполняет запросы и сразу показывает результат работы. Вложенный SQL — это язык, используемый в составе другого, например, C++ или Delphi. Разницы в синтаксисе практически нет, отличие только в том, как использовать запросы. В этой книге мы будем подразумевать, что у нас интерактивный SQL. Интерактивный SQL ближе к стандартному, а во вложенном очень часто встречаются отклонения и дополнения (например, в описании параметров, которые передаются запросу).

В стандартном SQL различаются только два типа данных: строки и числа, но некоторые производители добавляют свои типы, и MS SQL Server, который будет рассматриваться в книге, не исключение. Базы данных MS SQL Server поддерживают множество различных типов, что упрощает разработку, но нам необходимо научиться правильно отражать эти данные.

Числа в SQL делятся на два типа: целые (`INTEGER` или `INT`) и дробные (`DECIMAL` или `DEC`). Во многих базах данных числа делятся и по размеру используемого в базе пространства, но стандарт этого уже не требует. Строки в стандарте ограничены 254 символами, но в реальности размер намного больше.

Целочисленные типы указываются как есть, без каких-либо кавычек. Строковые типы, даты, глобально уникальные идентификаторы оформляются в одинарных кавычках, например:

```
'Это строка'
```

Если необходимо указать, что строка должна быть закодирована Unicode-символами, то необходимо перед строкой указать букву N, например:

```
N'Это пример Unicode строки'
```

Постепенно мы познакомимся с типами данных более подробно, а список поддерживаемых сервером MS SQL типов можно увидеть в *приложении 1*.

Если исходить из стандарта, то каждый запрос должен заканчиваться символом точки с запятой. Например:

```
Текст запроса;
```

Мы же в примерах этот символ будем очень часто опускать, потому что он не является обязательным для MS SQL Server. Я явно буду указывать на те случаи, когда символ просто необходим.

Работа с запросами

Для работы с первыми двумя главами вам понадобится база данных, которая поддерживает стандарт SQL 1992 года. Это практически единственное требование, но не все базы данных четко следуют стандарту. Желательно, чтобы база данных, которую вы выберете, поддерживала его максимально полно.

Лучшим вариантом будет MS SQL Server, потому что он максимально полно соответствует стандарту, но можно использовать и MS Access, хотя тут и есть отличия при установке связей между таблицами, и поддерживается только SQL, а не Transact-SQL.

Начиная с *главы 3*, мы будем углубляться в расширение языка SQL под названием Transact-SQL, и вот тут уже сервер баз данных MS SQL Server окажется незаменимым, потому что только он поддерживает это расширение.

Для вашего удобства на компакт-диске, прилагающемся к книге, в каталогах ChapterX (где X — номер главы) расположены файлы с текстом запросов и комментариями. Эти файлы избавят вас от необходимости набирать коды запросов на клавиатуре.

Все возможности языка SQL мы будем обсуждать с одновременным рассмотрением примеров. Сначала я буду приводить общий вид команды, а потом постепенно будем изучать возможности команды на примерах, максимально приближенных к реальным. Это позволит вам на практике увидеть, как использовать те или иные возможности, и иметь готовый багаж запросов для решения наиболее часто встречающихся задач.

Именование

Для того чтобы вам проще было понимать материал книги, вы должны знать, каким образом я присваиваю имена объектам базы данных. К каждому имени объекта я добавляю префикс, который будет указывать тип. Это позволит быстро определить по имени тип любого объекта. В *приложении 1* показаны типы полей, поддерживаемые в MS SQL Server. Например, если объект имеет тип `varchar`, то к имени объекта я добавляю префикс `vc`. После этого идет имя, отражающее суть объекта. Если этот объект предназначен для хранения номера телефона, то полное название поля будет `vcTelephon`.

В базах есть определенные поля, которые должны управляться по-другому. Например, для именования ключевого поля можно использовать следующий вид: `idИмяТаблицы`. Таким образом, по ключу можно понять, к какой таблице он относится, а по имени таблицы узнать имя ключа. В некоторых простых примерах для первичного ключа мы будем использовать просто имя `id`,

но при рассмотрении сложных связей я буду следовать правилу именования `idИмяТаблицы`.

Когда ключ используется для связи с другой таблицей, то после `id` нужно ставить имя таблицы, с которой происходит связь. Помимо этого, можно отделить идентификатор `id` знаком подчеркивания, чтобы сразу было видно, что это не основной ключ, а связь. Таким образом, по имени поля можно узнать, для связи с какой таблицей он предназначен.

Отдельного разговора заслуживают триггеры. Триггер — это процедура, которая вызывается в ответ на определенные действия над данными: вставку, изменение или удаление (операции `insert`, `update` или `delete`). Это как бы обработчики событий, в которых можно повлиять на обработку данных. Вот тут для именования наилучшим способом будет следующий:

```
ИмяТаблицы_СмыслТриггера_Операции
```

Порядок может быть любым, лишь бы вам было удобно, но желательно, чтобы все три составляющие присутствовали. Допустим, что у вас должен быть триггер на таблице `Person`, который при добавлении новых записей или изменении существующих должен проверять существование дублирующих записей. Имя такого триггера может быть следующим:

```
Person_CheckDouble_iu
```

По этому имени можно сразу понять, для какой таблицы он предназначен, что делает и когда выполняется. В MS SQL Server триггеры привязаны к базам данных и таблицам, а в Oracle все триггеры находятся в одном хранилище, вне зависимости от таблиц, и здесь вы реально оцените выгоду от такого метода именования.

К именам процедур и функций я не предъявляю особых требований, но рекомендую выделять их из общей массы. Какой способ выберете вы, зависит от личных предпочтений. В данной книге будут использоваться различные способы именования, без жесткой привязки к определенному правилу, чтобы показать вам все способы именования, но определенную закономерность легко проследить.

Язык SQL нечувствителен к регистру, а значит, если написать какой-то оператор в верхнем регистре и в нижнем, результат будет одинаковым. Например, слова `SELECT` и `select` для сервера являются идентичными.

Все операторы SQL я пишу в верхнем регистре. Это не является обязательным, так что можно все писать в нижнем регистре. Большинство интерпретаторов SQL-команд нечувствительны к регистру. Благодаря верхнему регистру вы сразу можете выделить операторы языка SQL из общей массы имен полей, таблиц и т. д.

CyD SQL Factory

Прежде чем мы начнем рассматривать сам язык, давайте посмотрим, как можно выполнять запросы в MS SQL Server и MS Access, чтобы вы могли на практике проверять все, что описывается в книге. Для работы с запросами я рекомендую использовать программу CyD SQL Factory, которую можно найти на прилагаемом компакт-диске в каталоге Programs. Программа не требует установки, просто скопируйте ее на свой локальный жесткий диск и используйте. Посмотрим, как это делать. Сначала нужно создать новый проект. Для этого выбираем меню **File | New project** (Файл | Новый проект). Программа создаст клиентское окно, которое разделено на две части. В левой половине находится текстовый редактор для ввода текста запросов, в правой половине можно увидеть схему базы данных (рис. П1). Мы пока не указали базу данных, которую хотим использовать, поэтому схема пуста.

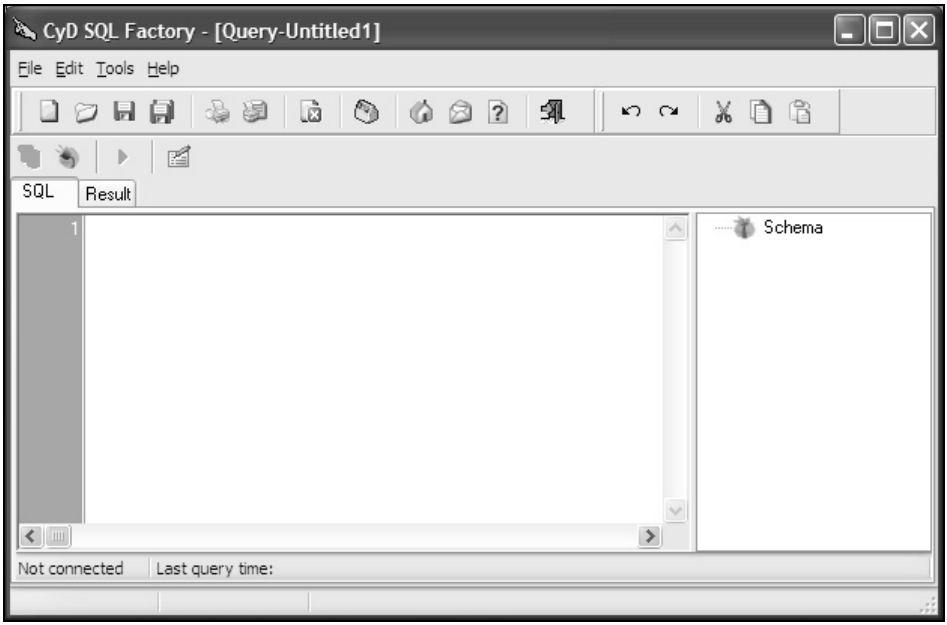

Рис. П1. Главное окно программы SQL Factory

Клиентское окно программы также разделено на две закладки: **SQL** и **Result** (Результат). На закладке **SQL** мы пишем запросы, а на закладке **Result** можно увидеть результат выполнения.

Для создания соединения с сервером нажмите кнопку **Connect to server** (Соединиться с сервером) в панели задач клиентского окна. Перед вами откроется окно, как на рис. П2.

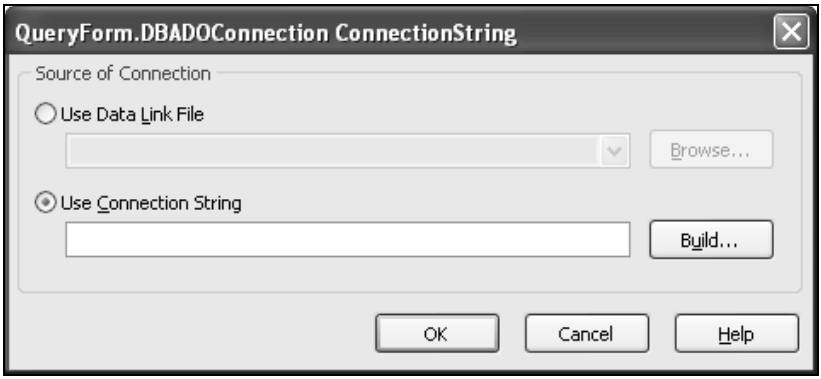

Рис. П2. Окно создания соединения с сервером

Соединение можно настроить, используя файл подключения (**Use Data Link File**) или строку подключения (**Use Connection String**). Рассмотрим второй вариант, как более простой и универсальный. Напротив строки **Use Connection String** нажмите кнопку **Build**, и на экране появится окно настройки строки соединения. На первой закладке нужно выбрать драйвер, который будет использоваться для подключения. Для MS Access нужно выбрать Microsoft Jet 4.0 OLE DB Provider (версия может отличаться), а для MS SQL Server лучше всего использовать Microsoft OLE DB Provider for SQL Server.

Теперь переходим на вкладку **Connection** (Соединение). Ее содержимое зависит от выбранного драйвера. В случае использования MS Access на этой вкладке будет только поле для указания пути к MDB-файлу базы данных. Для MS SQL Server это окно содержит больше настроек (рис. П3):

- **Select or enter a server name** (Выберите или введите имя сервера) — здесь нужно указать имя SQL-сервера, с которым вы хотите работать;
- **Enter information to log on to the server** (Введите информацию для входа на сервер) — здесь можно выбрать один из переключателей:
  - **Use Windows NT Integrated security** (Использовать интегрированную в Windows NT безопасность) — если вы вошли в ОС под учетной записью, которая имеет право доступа к SQL-серверу, то можно выбрать этот пункт. Например, SQL-сервер установлен локально, и вы входите в систему как администратор. Локальный администратор по умолчанию имеет право доступа к SQL-серверу, и можно выбирать этот переключатель;
  - **Use a specific user name and password** (Использовать указанные имя пользователя и пароль). В этом случае пароль будет указываться в соответствующих полях под этим переключателем;

- **Select the database on the server** (Выберите базу данных на сервере) — здесь можно выбрать из списка интересующую вас базу данных.

Рис. П3. Вкладка **Connection** при подключении к MS SQL Server

Если вы явно указывали имя пользователя и пароль, то обязательно установите флажок **Allow saving password** (Позволять сохранять пароль). Укажите необходимые данные и нажмите кнопку **ОК** для сохранения изменений. Программа попытается соединиться с сервером, и если все прошло удачно, в строке состояния появится надпись **Connected** (Подключен), а в дереве схемы появятся существующие таблицы выбранной базы данных (рис. П4).

Теперь в редакторе можно писать запросы и выполнять их. Попробуйте сейчас выполнить запрос:

```
SELECT * FROM ИмяТаблицы
```

Параметр `ИмяТаблицы` нужно заменить на имя, уже существующее в базе данных. Можете выбрать любое имя из списка, отображаемого в схеме. Чтобы

выполнить запрос, нажмите клавишу <F5>. Окно переключится на вкладку **Result**, где будет отображена сетка с данными выбранной таблицы.

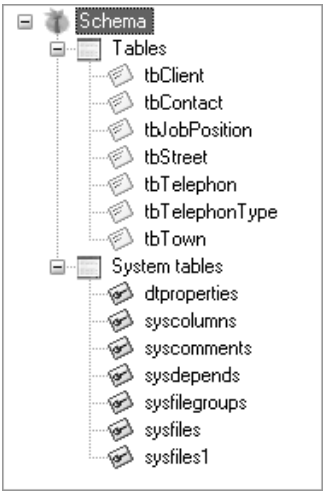

Рис. П4. Результат отображения схемы базы данных

Если у вас в редакторе большой запрос, но вы хотите выполнить только часть, выделите необходимый отрывок запроса и также нажмите клавишу <F5>.

Если вы выполнили запрос, который изменяет структуру базы данных (например, создает таблицу), то необходимо обновить схему, чтобы увидеть изменения. Для этого щелкните правой кнопкой в дереве схемы и в контекстном меню выберите пункт **Update Schema** (Обновить схему). С помощью этого же меню можно просмотреть любую таблицу. Выделите таблицу в схеме и затем в контекстном меню выберите пункт **Show All Table Data** (Отобразить все данные из таблицы).

Я рекомендую вам установить SQL Server с программой CyD SQL Factory и тестировать на своем сервере все запросы, описываемые в книге. Я уже не раз говорил, что практика является наилучшим учителем. Когда вы сами выполните все действия и увидите результат, то лучше поймете описываемый материал и запомните его. Я всегда это рекомендую своим читателям.

Query Analyzer

Программа CyD SQL Factory удобна тем, что способна работать с любыми базами данных, для которых есть Windows-драйвер, но ее возможности ограничены. Если вы будете работать с MS SQL Server, то более эффективным

решением будет использовать утилиту Query Analyzer, которая входит в поставку данного сервера. Давайте рассмотрим основные возможности работы с программой.

Запустите Query Analyzer, и откроется окно, как на рис. П5. Здесь необходимо задать параметры подключения к серверу. В выпадающем списке SQL Server необходимо задать имя сервера, к которому вы хотите подключиться. Если щелкнуть на кнопке справа от списка, то появится окно выбора доступных в сети серверов, но это окно далеко не всегда показывает все серверы, поэтому вы должны уметь вводить имя вручную.

Рис. П5. Окно создания соединения с сервером

Если у вас SQL-сервер установлен на локальном компьютере, то достаточно ввести точку или написать сетевое имя компьютера (можно вместо имени указать IP-адрес). Если сервер был установлен как экземпляр по умолчанию, то этого достаточно. Если вы устанавливали именованный экземпляр SQL Server, то вы должны написать имя в формате:

```
Имя компьютера\имя экземпляра
```

У меня компьютер называется notebook, а имя установленного экземпляра MS SQL Server — `flenov`, поэтому адрес подключения будет выглядеть так:

```
notebook\flenov
```

Чуть ниже необходимо выбрать параметры авторизации. Если вашей учетной записи, под которой вы сейчас работаете, разрешено подключаться к серверу SQL, то можно выбрать **Windows Authentication** (Аутентификация Windows), иначе нужно выбрать **SQL Server Authentication** (Аутентификация SQL Server) и указать явное имя и пароль, которым разрешено подключение к серверу.

Если установить флажок **Start SQL Server if it is stopped** (Запустить SQL Server, если он остановлен), то программа может запустить сервер баз данных, если он в данный момент не запущен. Это очень удобно, если вы не установили MS SQL Server для автоматического запуска. У меня SQL-сервер работает на ноутбуке, и я программирую в нем не каждый день, поэтому, чтобы запуск сервера не отнимал лишнее время при загрузке ОС, я отключил автозагрузку MS SQL Server, и эта опция очень сильно помогает мне.

Если вы указали параметры подключения верно, то после нажатия кнопки **ОК** перед вами появится окно программы для работы с SQL-запросами (рис. П6). Это окно разделено вертикально на две части. В верхней части окна мы пишем SQL-запросы, а в нижней отображается результат работы.

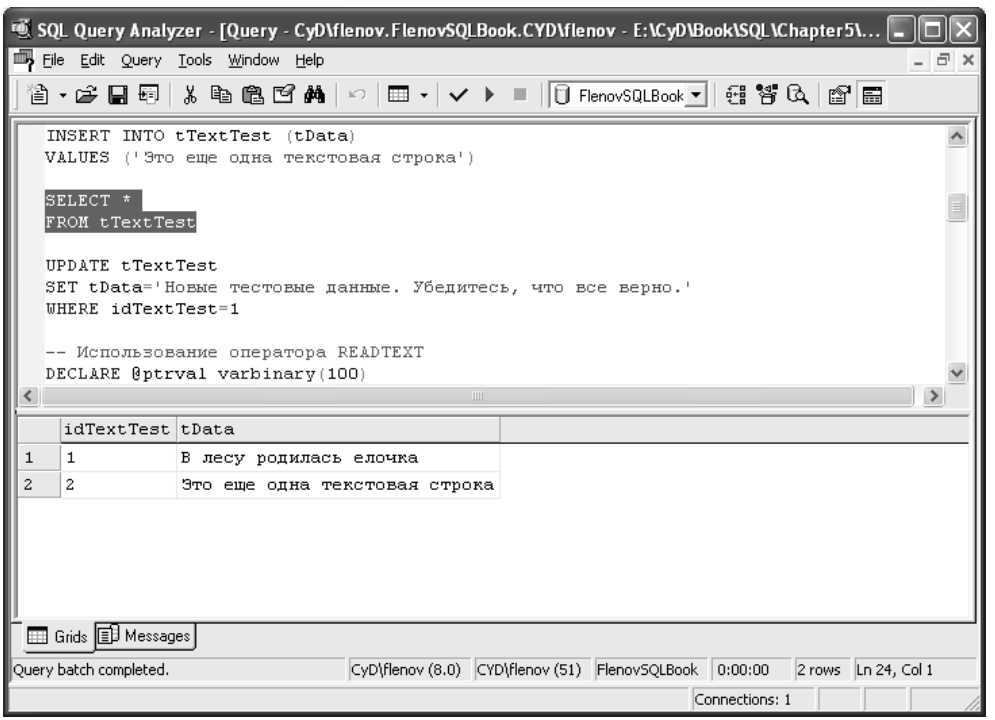

Рис. П6. Окно работы с SQL-запросами

Чтобы Query Analyzer выполнил запрос, нажмите клавишу <F5>. При этом будет выполнен весь код, набранный в окне запроса. Если необходимо выполнить только часть, то перед нажатием <F5> выделите необходимую часть запроса.

Если вы набрали сразу несколько запросов, то при запуске на выполнение сервер будет выполнять их как одно целое. Иногда это может привести

к ошибкам. Например, если первый запрос создает объект (таблицу, процедуру, функцию и т. п.), а второй обращается к объекту. Если оба запроса будут выполняться как одно целое, то сервер не сможет найти создаваемый объект. Чтобы решить эту проблему, необходимо написать между запросами оператор `GO`:

```
Код создания объекта
GO
Код обращения к объекту
```

Теперь два запроса будут отправлены серверу по отдельности, и они будут выполнены корректно.

Иногда нам придется переключаться между различными базами данных. Для этого в Query Analyzer можно использовать выпадающий список в панели инструментов или набрать команду:

```
USE Имя базы
```

Команды `USE` и `GO` не относятся к операторам Transact-SQL, а выполняются в клиентской программе, поэтому могут не действовать в других программах работы с запросами.

На этом мы пока оставим Query Analyzer, а его дополнительные возможности будем осваивать по мере надобности. А теперь уже пора перейти к знакомству непосредственно с операторами ANSI SQL и Transact-SQL.

Глава 1

Управление базой данных

Большинство авторов начинает рассматривать SQL с операторов получения данных из базы. Но у нас еще нет баз данных и неоткуда брать данные. Поэтому мы начнем с создания базы данных и таблиц. Когда у нас будет готова тестовая база, мы заполним ее данными и тогда уже научимся работать с ними.

Итак, в этой главе нам предстоит узнать:

- как создавать базу данных с помощью SQL-запросов;
- как изменять параметры базы данных;
- как создавать таблицы;
- как изменять параметры таблицы.

Я рекомендую выполнить все сценарии из каталога Chapter1 на компакт-диске, прилагаемом к книге. Это позволит вам иметь готовую структуру базы данных, на которой можно будет тестировать запросы из следующих глав.

Большинство описываемых в этой главе операторов SQL будут работать на большинстве баз данных. Но есть некоторые тонкости. Например, в MS Access нельзя создавать базу данных, потому что в этом случае база данных — файл, который создается с помощью одноименной программы. В других базах данных операторы создания баз и таблиц имеют точно такой же синтаксис, но могут быть отличия в поддерживаемых параметрах из-за большего или меньшего количества возможностей.

Операторы по описанию объектов базы данных выделяют в отдельный язык (подъязык SQL) — DDL (Data Definition Language, язык определения данных). Именно он будет рассматриваться в этой главе, ведь нам предстоит научиться описывать данные.

1.1. Создание и удаление базы данных

Информация о каждой базе данных в SQL Server хранится в таблице `sysdatabases` базы данных master. Поэтому желательно (но не обязательно) использовать базу данных master во время создания базы. К тому же, после изменения любой пользовательской базы данных нужно создавать резервную копию базы данных master. О резервном копировании и восстановлении мы поговорим в *разд. 4.10*.

Создание базы данных — это процесс указания имени, размера и расположения файлов.

В Transact-SQL для создания базы данных есть команда `CREATE DATABASE`. Эта команда может выполняться только с сервером SQL Server. При использовании базы данных MS Access команда недоступна, потому что базой данных является файл с расширением mdb, который создается в программе Access, и к которому мы подключены.

Сервер MS SQL может содержать несколько баз данных. Вы можете подключиться к любой из них (системной или тестовой, которые присутствуют в стандартной поставке) и создать новую базу данных, но желательно подключиться к базе данных master.

Синтаксис команды создания базы данных показан в листинге 1.1.

Листинг 1.1. Создание базы данных

```
CREATE DATABASE имя
[ ON
    [ < filespec > [ ,...n ] ]
    [ , < filegroup > [ ,...n ] ]
]
[ LOG ON { < filespec > [ ,...n ] } ]
[ COLLATE имя_раскладки ]
[ FOR LOAD | FOR ATTACH ]

< filespec > ::=
[ PRIMARY ]
( [ NAME = логическое_имя_файла , ]
    FILENAME = 'имя_файла_в_ОС'
    [ , SIZE = размер ]
```

```
    [ , MAXSIZE = { максимальный_размер | UNLIMITED } ]
    [ , FILEGROWTH = увеличение ] ) [ ,...n ]

< filegroup > ::=
FILEGROUP файловая_группа < filespec > [ ,...n ]
```

Давайте разберем этот код подробно. Первая строчка содержит имя команды и параметр:

```
CREATE DATABASE имя
```

В качестве параметра выступает имя создаваемой базы данных.

Если вы посмотрите на остальные параметры команды, то заметите, что все они находятся в квадратных скобках. Обычно в квадратных скобках указываются необязательные параметры. Получается, что самый простой вариант команды создания базы выглядит так:

```
CREATE DATABASE Имя
```

Имена, состоящие из нескольких слов, необходимо заключать в квадратные скобки:

```
CREATE DATABASE [Тестовая база]
```

Очень интересной является следующая строчка:

```
[ FOR LOAD | FOR ATTACH ]
```

Здесь в квадратных скобках указано два значения, разделенных вертикальной чертой. Это значит, что они являются необязательными, а вертикальная черта соответствует слову "или", т. е. в запросе можно будет указывать или `FOR LOAD`, или `FOR ATTACH`, или вообще ничего. Оба параметра одновременно указывать нельзя.

В угловых скобках указываются имена секций. Например, в описании оператора `CREATE DATABASE` есть два указания на `< filespec >`. Эта секция может идти после ключевого слова `ON` и после `LOG ON`. Описание самой секции идет после:

```
< filespec > ::=
```

Если вы имеете опыт программирования на одном из высокоуровневых языков, то в секциях вы уже, наверное, увидели аналогию с процедурами. Название секции `< filespec >` аналогично имени процедуры, а после `< filespec > ::=` идет сам код процедуры.

Следующая интересная строчка:

```
[ < filespec > [ ,...n ] ]
```

Здесь < filespec > — описание файла, а [,...n] указывает на то, что возможно несколько описаний.

С помощью круглых скобок параметры объединяются в группу, например:

```
( [ NAME = логическое_имя_файла , ]
    FILENAME = 'имя_файла_в_ОС'
    [ , SIZE = размер ]
    [ , MAXSIZE = { максимальный_размер | UNLIMITED } ]
    [ , FILEGROWTH = увеличение ] ) [ ,...n ]
```

В данном случае в группу объединены параметры NAME, FILENAME, SIZE, MAXSIZE и FILEGROWTH, т. к. все они описывают файл. Обязательным является только параметр FILENAME. После круглых скобок идет [,...n], это означает, что может быть несколько описаний файлов (для каждого файла базы данных свое описание).

Параметр FILENAME интересен еще и тем, что его значение задается с помощью знака равенства, после которого идет текст в одинарных кавычках:

```
FILENAME = 'имя_файла_в_ОС'
```

Кавычки в данном случае обязательны. По наличию кавычек достаточно просто определить тип параметра. Если они присутствуют, то параметр строковый, иначе числовой. Например, параметр SIZE не содержит кавычек, а значит, он числовой:

```
SIZE = Размер
```

На первый взгляд, общий вид оператора CREATE DATABASE достаточно сложен, но здесь не так уж и много параметров. Давайте рассмотрим основные, а потом увидим их действие на практике.

- PRIMARY. Этот параметр указывает файл в основной файловой группе. Эта файловая группа содержит все системные базы данных. Она также содержит все объекты, не назначенные другим файловым группам. Каждая база данных содержит один основной файл данных. Основной файл — это стартовая точка базы данных. Кроме того, он указывает на ее местонахождение. Для основного файла рекомендуется расширение mdf. Если вы не укажете этот параметр, первый файл списка описания будет использован как основной.
- FILENAME. Этот параметр указывает имя и путь к файлу в операционной системе. Путь должен указывать папку на сервере, где установлен SQL Server. Нельзя использовать сетевые диски с других компьютеров.
- SIZE. Этот параметр указывает размер файла данных или журнала. Вы можете указать размер в мегабайтах (значение по умолчанию) или в кило-

байтах. Минимальный размер — 512 Кбайт для обоих файлов: журнала транзакций и файла данных. Размер, указанный для основного файла базы данных, должен быть больше или равен размеру основного файла базы данных model. Мы уже говорили, что база model копируется во все новые базы данных, поэтому размер новой не может быть меньше размера model, иначе копирование станет невозможным. Когда вы добавляете новый файл базы данных или журнала без указания размера, то сервер использует значение размера по умолчанию 1 Мбайт.

- `MAXSIZE`. Этот параметр указывает максимальный размер, до которого файл может увеличиваться. Вы можете указать размер в мегабайтах (значение по умолчанию) или в килобайтах. Если вы не укажете максимальный размер, файл будет увеличиваться, пока диск не будет заполнен целиком.
- `FILEGROWTH`. Этот параметр указывает размер приращения файла. Значение этого параметра для файла не может превышать значение `MAXSIZE`. Значение 0 указывает на запрет увеличения. Значение может быть указано в мегабайтах (по умолчанию), килобайтах или процентах. Значение по умолчанию, если этот параметр не указан, — 10%, а минимальный размер — 64 Кбайт. Указанный размер округляется до ближайшего числа, кратного 64 Кбайт.
- `COLLATE`. Этот параметр указывает значение по умолчанию для сопоставления в базе данных. Сопоставления (кодировка или раскладка) включают правила, контролирующие использование символов для языка и алфавита.

Во время создания базы данных очень важно понимать, как SQL Server хранит данные, чтобы вы могли сосчитать и указать размер дискового пространства для размещения базы данных. Во время создания баз учитывайте следующее.

- Все базы данных имеют основной файл данных, определяемый именем файла с расширением mdf, и один или более файлов журнала, определяемый именем файла с расширением ldf. База данных может также иметь вторичные файлы данных, которые определяются по имени файла с расширением ndf. Файлы могут объединяться в группы, о чем мы поговорим в *разд. 1.1.1*.
- Физические файлы имеют двойное именование — имя ОС и имя, которое вы можете использовать в операторах Transact-SQL (логическое имя, которое указывается в параметре `NAME`).
- Когда вы создаете базу данных, в нее копируется содержимое базы данных model, которая включает в себя системные таблицы и может содер-