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.

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

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-

жать пользовательские таблицы. Минимальный размер создаваемой базы данных должен быть равен или больше размера базы данных model.

- Сервер SQL хранит, читает и записывает данные блоками по 8 Кбайт, эти блоки называются страницами. База данных может хранить 128 страниц на мегабайт. Все страницы хранятся в пространстве. Пространство — это 8 последовательных страниц, или 64 Кбайт. Поэтому база данных имеет 16 пространств в мегабайте.

Страницы и пространства — это основа структуры базы данных SQL Server. Сервер MS SQL использует различные типы страниц, некоторые следят за выделенным пространством, а некоторые содержат пользовательские данные и индексы. Страницы, которые отслеживают выделенное пространство, содержат плотно сжатую информацию. Это позволяет серверу MS SQL эффективно помещать их в память, чтобы облегчить просмотр.

Сервер SQL использует два типа пространств.

- Пространства, которые хранят страницы от двух и более объектов, называются смешанными. Каждая таблица начинается как смешанное пространство. Вы используете смешанное пространство, главным образом, для страниц, которые хранят пространство и содержат маленькие объекты.
- Пространства, которые хранят все 8 страниц, выделенных одному объекту, называются однородными пространствами. Они используются, когда таблице или индексу требуется более 64 Кбайт.

Первое пространство для каждого файла является смешанным и содержит страницы заголовка файла, а потом выделяется по три страницы непосредственно для данных. Сервер выделяет эти смешанные пространства, когда вы создаете основной файл данных, и использует эти страницы для своих внутренних задач. Страница заголовка файла содержит атрибуты файла, такие как имя базы данных, которая хранится в файле, файловая группа, минимальный размер, размер приращения. Это первая страница в каждом файле (Страница 0).

Страница свободного пространства (PFS) — это выделенная страница, содержащая информацию о свободном пространстве, доступном в файле. Эта информация хранится в странице 1. Каждая такая страница может простираться на 8000 смежных страниц, что приблизительно составляет 64 Мбайт данных.

Журнал транзакций захватывает всю необходимую информацию о происходящих на сервере изменениях для восстановления базы данных в момент возникновения системной ошибки и для обеспечения целостности данных. О журнале мы будем говорить чуть позже, а сейчас необходимо понимать, что журнал — это отдельный файл, который требует дискового пространства.

Теперь рассмотрим, как можно удалять созданную базу данных:

```
DROP DATABASE имя
```

Нельзя удалять:

- базу данных, которая открыта для чтения или записи любым пользователем, поэтому при удалении вы также не должны быть к ней подключены. Лучше всего подключиться к базе данных master;
- базу данных, которая опубликовала любую свою таблицу как часть репликации SQL Server;
- системную базу данных.

Рассмотрим простейший пример создания и удаления базы данных. Для создания базы данных достаточно написать:

```
CREATE DATABASE имя
```

Все остальные параметры являются необязательными. Попробуем создать базу данных с именем `TestDatabase` и удалить. Сначала создадим базу:

```
CREATE DATABASE TestDatabase
```

И тут же ее удалим:

```
DROP DATABASE TestDatabase
```

Эти команды нужно выполнять по отдельности. Одновременно выполнять нельзя, иначе сервер вернет сообщение об ошибке. Рекомендую тестировать примеры. После этого тестовые базы данных можно удалять.

Теперь посмотрим, какие еще возможности дает нам команда создания базы данных. Имя базы данных ограничено 128 символами, если вы явно указываете логическое имя файла журнала. Я считаю, что этого вполне достаточно. Если логическое имя журнала не задано, то размер сокращается до 123 символов. Это связано с тем, что логическое имя журнала также ограничено 128 символами, и если оно не указано, то в качестве имени используется имя базы плюс суффикс `_log`. Самое интересное, что суффикс занимает четыре символа, а 128 – 4 = 124. Почему Microsoft ограничивает длину имени числом 123, для меня остается загадкой. Быть может, составители документации разучились считать?

Рассмотрим основные правила, которым должны подчиняться имена баз данных и объектов в ней.

- Первый символ должен быть буквой a—z, A—Z.
- После первого символа может идти буква, цифра или символы _, @ или #.
- Идентификаторы, начинающиеся с символов, имеют специальное назначение:
  - идентификаторы, начинающиеся с символа @, являются локальными переменными или параметрами;

- идентификаторы, начинающиеся с символа #, являются временными таблицами или процедурами;
- идентификаторы, начинающиеся с символа ##, являются глобальными временными объектами.

Эти правила относятся ко всем именам — базы данных, таблиц, полей и т. д. С некоторыми из них мы еще будем не раз встречаться на практике, например, при создании временных объектов.

Чтобы указать дополнительные параметры, нужно после оператора `CREATE DATABASE` написать `ON` и далее в круглых скобках перечислить параметры:

```
CREATE DATABASE TestDatabase ON
(
 Параметры
)
```

Если какие-то параметры не заданы, то их значения берутся по умолчанию. Пока мы не будем обсуждать возможные параметры, потому что они одинаковы для базы данных и журнала транзакций, и нужно рассмотреть, как описывается журнал.

Журнал транзакций — это файл, в котором сохраняется информация об активности в отношении базы данных. Например, вы выдаете запрос на обновление данных. В общем виде сервер выполняет следующие действия:

- ❒ если явно не указано, то сервер автоматически создает новую транзакцию;
- ❒ запрос сохраняется в журнале;
- ❒ сервер начинает выполнять обновление;
- ❒ если все изменения прошли успешно, то транзакция помечается как завершенная, и данные считаются обновленными;
- ❒ если произошел сбой сервера или ошибка обработки данных, то транзакция не завершается. В этом случае все изменения, сделанные незавершенной транзакцией, отменяются, как будто ничего не произошло. Если во время транзакции отключилось питание и данные не успели обновиться, то при следующей загрузке сервер увидит в журнале незавершенную транзакцию и попытается завершить.

Получается, что журнал обеспечивает целостность данных. Посмотрим на файловую систему NTSF. Это в своем роде тоже база данных. Если какой-либо файл поврежден и не читается, то он только нарушает целостность файловой системы и его необходимо удалить. Программа сканирования scandisk проверяет все транзакции с файловой системой и быстро находит незавершенные операции, отменяя их или завершая.

Вы должны уделять внимание не только корректности и оптимальности создания базы данных, но и журналу транзакций.

Теперь давайте рассмотрим, какие параметры можно указывать при создании файлов базы данных и журнала транзакций:

- `NAME` — логическое имя, которое будет отображаться в SQL-сервере;
- `FILE NAME` — физическое расположение и имя файла;
- `SIZE` — начальный размер файла. Это начальное значение, которое может увеличиваться по мере надобности;
- `MAXSIZE` — максимальный размер файла. Чтобы файл не имел ограничений, необходимо указать в этом параметре `UNLIMITED` вместо реального значения. Но я рекомендую указывать такое значение, которое не сможет переполнить весь жесткий диск;
- `FILEGROWTH` — приращение, т. е. на сколько должен увеличиться размер файла, если текущего размера недостаточно. Приращение можно указывать как в реальных значениях (мегабайты), так и в процентном отношении к текущему размеру. Приращение должно быть достаточным, чтобы эта операция выполнялась как можно реже.

Давайте теперь рассмотрим несколько примеров, которые позволят закрепить все вышесказанное. В листинге 1.2 описываются параметры файла данных и параметры журнала. Если какой-то параметр не указан явно, то его значение берется по умолчанию.

Листинг 1.2. Создание базы данных с описанием параметров файлов

```
CREATE DATABASE TestDatabase
ON
(
  NAME = TestDatabase_data,
  FILENAME = 'c:\data\test.mdf',
  SIZE = 10MB,
  MAXSIZE = 100GB,
  FILEGROWTH = 5MB
 )
LOG ON
(
  NAME = 'TestDatabase_log',
  FILENAME = 'c:\data\test.ldf',
```

```
  SIZE = 5MB,
  MAXSIZE = 10GB,
  FILEGROWTH = 5%
)
```

Посмотрим, что делает этот сценарий. В первой строчке мы указываем, что необходимо создать базу данных с именем `TestDatabase`. Затем идет ключевое слово `ON` и в круглых скобках перечисляются параметры файла базы данных. Мы указываем пять параметров: логическое имя (`NAME`), физическое расположение файла данных (`FILENAME`), начальный размер (`SIZE`) 10 Мбайт, максимальный размер (`MAXSIZE`) 1000 Гбайт и приращение (`FILEGROWTH`) 5 Мбайт. Этого достаточно только для тех баз данных, где добавление новых записей происходит редко.

Для часто изменяемых баз я рекомендую указывать большое приращение для предотвращения частого автоувеличения. Это облегчает решение административных задач, уменьшает фрагментацию файла и избавляет сервер от лишнего расширения файлов базы. Приращение относится как к файлу базы данных, так и к файлу журнала.

Если база данных не изменяется в размере (не добавляются новые данные), но очень часто происходит изменение уже существующих данных, то приращение для файла данных можно сделать небольшим (5—10 Мбайт, в зависимости от размера одной записи таблицы). При этом файл журнала должен иметь большее приращение, которое лучше всего указать в процентах от существующего размера. Исходя из практики, я бы порекомендовал значение 10%, но в зависимости от задачи значение можно скорректировать.

Если данные базы используются только как справочник и изменяются очень редко, то размер приращения базы и журнала можно сделать минимальным. Запросы на выборку данных не сохраняются в журнале транзакций, поэтому можно ограничиться увеличением в 1 Мбайт. Но при этом значение приращения желательно сделать больше, чем потребуется для хранения 100 строк данных. Размер строки можно примерно рассчитать по размеру всех полей самой большой таблицы.

Если вы используете автоматическое увеличение, то лучше всего указать максимальный размер. Это позволит вам предотвратить заполнение базой данных всего жесткого диска.

Приращение для файла данных можно указывать и в процентах, например:

```
CREATE DATABASE TestDatabase
ON
(
```