**Цели работы:**

* научиться редактировать базу данных, таблицы, связи между таблицами средствами MS SQL Server Management Studio, а также средствами языка T-SQL;
* изучить возможности оператора ALTER.

**Теоретическая часть:**

Язык Transact-SQL поддерживает изменение структуры следующих объектов базы данных:

* базы данных;
* таблицы;
* представления;
* схемы;
* хранимые процедуры;
* триггеры.

**Изменение базы данных**

Для изменения физической структуры базы данных используется инструкция **ALTER DATABASE**.

Язык Transact-SQL позволяет выполнять следующие действия по изменению свойств базы данных:

* добавлять и удалять один или несколько файлов базы данных;
* добавлять и удалять один или несколько файлов журнала;
* добавлять и удалять файловые группы;
* изменять свойства файлов или файловых групп;
* устанавливать параметры базы данных;
* изменять имя базы данных с помощью хранимой процедуры sp\_rename.

Базовый синтаксис:

ALTER DATABASE database

{ ADD FILE < filespec > [ ,...n ] [ TO FILEGROUP filegroup\_name ]

| ADD LOG FILE < filespec > [ ,...n ]

| REMOVE FILE logical\_file\_name

| ADD FILEGROUP filegroup\_name

| REMOVE FILEGROUP filegroup\_name

| MODIFY FILE < filespec >

| MODIFY NAME = new\_dbname

| MODIFY FILEGROUP filegroup\_name {filegroup\_property | NAME = new\_filegroup\_name }

| SET < optionspec > [ ,...n ] [ WITH < termination > ]

| COLLATE < collation\_name >

}

где **database** – имя, базы данных, которую надо изменить. Далее построчно идут параметры или свойства базы данных, которые нужно изменить.

**Изменение файла БД**

Когда данные увеличиваются или увеличивается активность изменения данных, вам может понадобиться увеличить размер файлов данных или журнала. Для увеличения размер файлов можно использовать программу управления базой данных SQL Server Enterprise Manager, которая обладает визуальным интерфейсом или воспользоваться оператором ALTER DATABASE.

Вы можете контролировать размер базы данных с помощью:

* конфигурирования файлов базы данных и журнала на автоматическое приращение;
* увеличивать или уменьшать текущее значение максимального размера файлов базы данных и журнала вручную;
* вручную добавлять дополнительные файлы журнала или данных.

Вы можете установить автоматическое приращение с помощью оператора ALTER DATABASE. Использование автоматического приращения уменьшает администраторские задачи, заключенные в ручном увеличении размера базы данных, но сложнее контролировать количество используемого сервером MS SQL Server пространства. Увеличение лучше всего контролировать самостоятельно.

Если вы не сконфигурировали существующие файлы на автоматическое приращение, вы все еще можете увеличить размер. Если указан ноль для значения автоматического увеличения, значит, автоматического увеличения нет.

Вы можете создать вторичные файлы базы данных, для расширения размера базы данных. Но лучше использовать дополнительные файлы базы данных, чтобы расположить файлы на отдельном физическом диске или выделить объект базы данных из общей массы.

Следующий пример увеличивает текущий размер журнала базы данных NewDB:

ALTER DATABASE NewDB

MODIFY FILE

(

NAME = 'NewDB',

SIZE = 100mb

)

Теперь попробуем добавить дополнительный файл. Да, и это возможно. Допустим, что свободное пространство на жестком диске, но у вас есть дисковый массив, который позволяет управлять массивом на горячую. Прямо во время работы вставляем новый диск, создаем на нем файл для базы данных и продолжаем спать спокойно. Следующий пример показывает, как можно добавить файл к базе данных Sample.

ALTER DATABASE Sample

ADD FILE

(

NAME = 'SampleData2',

FILENAME='C:\Temp\SampleData2.mdf',

SIZE = 15mb

)

Вот тут необходимо сделать одно замечание, если для переноса данных с одного сервера на другой захотите воспользоваться функциями отключения и подключения базы данных, то не забудьте скопировать в новое место все файлы.

Допустим, что вам необходимо произвести массовую загрузку данных. База данных на данный момент занимает 10 мегабайт, а приращение, указанное в настройках в случае нехватки места составляет 1 мегабайт. Если вы будете загружать еще 10МБ данных, то во время загрузки произойдет 10 увеличений по одному мегабайту файла данных. Это отнимет лишнее время, поэтому вполне логичным будет увеличить размер файла вручную.

Для ручного увеличения размера файла до 200Мб необходимо выполнить следующую команду:

ALTER DATABASE testdatabase

MODIFY FILE

(

NAME = testdatabase,

SIZE = 200MB

)

Вторая строка содержит команду MODIFY FILE, что означает изменения файла. В скобках необходимо указать имя файла, размер которого надо изменить в параметре NAME и новый размер. Новый размер обязательно должен быть больше текущего, потому что команда позволяет только увеличивать файл, но не уменьшать.

Если вы хотите узнать текущий размер файлов или не помните имена, выполните команду:

EXEC sp\_helpdb имя

где **имя** - имя базы данных. Например, следующая команда отображает информацию о базе данных «Поставки», в которую входит информация и о файлах:

EXEC sp\_helpdb Поставки

Помните, что таким образом вы можете только увеличивать размер файлов, но не уменьшать. Уменьшение файлов данных и журнала – это отдельная тема. Итак, если вы создали базу данных с файлом в 1 Мб, и потом увеличили ее размер до 5 мегабайт то вернуть изначальный размер командой ALTER DATABASE невозможно. Это значит, что следующий запрос вернет ошибку, если размер файла FlenovSQLBook больше 2Мб:

ALTER DATABASE FlenovSQLBook

MODIFY FILE

(

NAME = FlenovSQLBook,

SIZE = 1MB

)

С помощью команды ALTER DATABASE можно изменять и размер файла журнала, только в параметре NAME нужно указать имя файла журнала, который нужно увеличить:

ALTER DATABASE TestDatabase

MODIFY FILE

(

NAME = testdatabase\_log,

SIZE = 2MB

)

Для оптимальной производительности сервера и уменьшения количества увеличения файлов (это отнимает лишнее время и может затормозить работу сервера):

* выделяйте необходимое количество начального пространство, чтобы избежать частого автоматического приращения;
* указывайте максимальный размер файлов данных, если у вас есть несколько баз данных;
* устанавливайте размер приращения файлов данных и журнала так, чтобы не было частого автоматического увеличения.

Когда база данных увеличивается или когда увеличивается активность изменений, вам может понадобиться расширить журнал транзакций. Внимательно следите, чтобы журнал транзакций не оказался слишком маленьким. Регулярное наблюдение за журналом поможет вам определить оптимальное увеличение журнала и не допустить его переполнения. Если ваш журнал транзакций не может быть увеличен, то SQL Server не сможет записывать транзакции и не позволить делать изменения в базе данных.

Если данные изменяются достаточно часто, то необходимо выделить максимально возможное (но разумное) пространство и следить, чтобы всегда было достаточно свободного места в журнале даже для выполнения самой массовой операции обновления данных. Если база используется редко, и изменения происходят время от незначительно, то журнал можно выделить небольшим. Помните, что журнал транзакций пополняется только во время добавления, изменения и удаления данных. Во время выборки журнал не используется.

Некоторые ситуации, которые увеличивают активность журнала транзакций:

* загрузка информации в таблицу, которая содержит индексы. Сервер SQL записывает все вставки и изменение индексов. При загрузке таблиц без индексов, SQL Server записывает в журнал только расширение пространства.
* транзакции, которые выполняют много изменений (INSERT, UPDATE и DELETE) в таблице с помощью одной транзакции.
* добавление или изменение данных Image или Text.

**Добавление/удаление файла БД**

Когда вы используете автоматическое увеличение, и база данных состоит из нескольких файлов, SQL Server использует стратегию пропорционального заполнения между всеми фалами каждой файловой группы. По мере записи данных в файловую группу, сервер записывает результат пропорционально свободному пространству в каждом файле файловой группы и после этого записывает в следующий файл. Тут есть и положительные и отрицательные моменты. Положительное видно сразу, ведь если диски подключены к разным контроллерам, то мы добиваемся параллельности, а жесткие диски являются самым слабым звеном. Но отрицательный момент может возникнуть при автоматическом увеличении файлов. Чаще всего размеры дисков разные, да и свободное пространство на них распределяется не равномерно. Теперь, один диск будет заполнен раньше, чем другой. Нет, база данных в этом случае останется доступной, и запись будет только на один диск, но произойдет резкое падение производительности во время записи.

Добавление или удаление файлов базы данных осуществляется посредством инструкции ALTER DATABASE. Операция добавления нового или удаления существующего файла указывается предложением **ADD FILE** и **REMOVE FILE** соответственно. Кроме этого, новый файл можно определить в существующую файловую группу посредством параметра TO FILEGROUP.

Базовый синтаксис:

ALTER DATABASE SampleDb

ADD FILE (

NAME=sampledb\_dat1,

FILENAME='D:\sampledb1.mdf',

SIZE = 10,

MAXSIZE = 100,

FILEGROWTH = 5

)

В этом примере инструкция ALTER DATABASE добавляет новый файл с логическим именем sampledb\_dat1. Здесь же указан начальный размер файла 10 Мбайт и автоувеличение по 5 Мбайт до максимального размера 100 Мбайт. Файлы журналов добавляются так же, как и файлы баз данных. Единственным отличием является то, что вместо предложения ADD FILE используется предложение **ADD LOG FILE**.

Пример:

Допустим, что на вашем основном диске, где расположена база данных, заканчивается место на диске. В этом случае, если есть другие логические разделы или жесткие диски, можно добавить к серверу новый файл и расположить его на этом диске. Добавление чем-то похоже на определение файла при создании базы данных:

ALTER DATABASE TestDatabase

ADD FILE

(

NAME = dbFile2,

FILENAME = 'C:\Data\dbfile2.ndf',

SIZE = 5MB,

FILEGROWTH = 5MB

)

В данном примере изменяется уже существующая база данных, на которой мы тестировали не один пример - TestDatabase. Затем указывается ключевое слово ADD FILE, указывающее на необходимость добавить новый файл. После этого ключевого слова, в круглых скобках описываются параметры файла точно так же, как при описании файла во время создания базы данных.

В качестве расширения для файла выбрано .ndf. Такое расширение рекомендуется для всех вторичных файлов, но может быть и другим. Для основного файла рекомендуется указывать расширение .mdf.

Для удаления файла из базы данных совместно с ALTER DATABASE используется ключевое слово REMOVE FILE. Если вы создали файл и поместили в него таблицу, то файл уже нельзя удалить, пока он не станет пустым, то есть, пока из файла не будут удалены все таблицы.

Для удаления нужно указать оператор REMOVE FILE и имя удаляемого файла:

ALTER DATABASE TestDatabase

REMOVE FILE dbFile2

**Добавление/удаление файловых групп БД**

Для добавления файловой группы используется оператор ADD FILEGROUP. Например, следующий сценарий добавляет группу с именем fgNewGroup:

ALTER DATABASE TestDatabase

ADD FILEGROUP fgNewGroup

Теперь посмотрим, как можно добавить файл в эту группу:

ALTER DATABASE TestDatabase

ADD FILE

(

NAME = dbFile2,

FILENAME = 'c:\Data\dbfile2.ndf',

SIZE = 5MB,

FILEGROWTH = 5MB

)

TO FILEGROUP fgNewGroup

Принцип такой же, как и у добавления файла, но после скобок указывается TO FILEGROUP и имя группы, в которую добавляется файл.

Файловую группу нельзя удалять, если в ней есть файл. Сначала необходимо удалить этот файл, а потом уже удалять группу. Для удаления группы используется REMOVE FILEGROUP, например:

ALTER DATABASE TestDatabase

REMOVE FILEGROUP fgNewGroup

**Изменение свойств файлов и файловых групп БД**

С помощью предложения **MODIFY FILE** можно выполнять следующие действия по изменению свойств файла:

* изменять логическое имя файла, используя параметр *NEWNAME*;
* увеличивать значение свойства *SIZE*;
* изменять значение свойств *FILENAME*, *MAXSIZE* и *FILEGROWTH*;
* отмечать файл как *OFFLINE*.

Подобным образом с помощью предложения **MODIFY FILEGROUP** можно выполнять следующие действия по изменению свойств файловой группы:

* изменять логическое имя файловой группы, используя параметр NAME;
* помечать файловую группу, как файловую группу по умолчанию, используя для этого параметр DEFAULT;
* помечать файловую группу как позволяющую осуществлять доступ только для чтения или для чтения и записи, используя для этого параметр read\_only или read\_write соответственно.

**Переименование БД**

Иногда бывает необходимость переименовать базу данных. В моей практике это очень редко приходилось делать, но все же. Переименовать можно с помощью оператора MODIFY NAME. Например, следующий сценарий изменяет имя базы данных TestDatabase на MyDatabase:

ALTER DATABASE TestDatabase

MODIFY NAME = MyDatabase

При этом вы не должны быть подключены к этой базе данных, лучше всего, если подключение будет к базе данных master. Если к базе данных, которую необходимо переименовать будет подключен хоть один пользователь, то переименование не сможет быть выполнено.

**Изменение свойств БД**

У базы данных существует множество свойств, которые мы не задавали во время создания, но которые можно изменить уже у существующей базы. К таким свойствам относятся уровень доступа, модель восстановления и т.д. Давайте рассмотрим, что и как можно изменять.

Для изменения свойства используется оператор SET. Команда будет выглядеть следующим образом:

ALTER DATABASE Имя\_базы

SET имя\_свойства

После ALTER DATABASE указывается имя базы данных, свойства которой нужно изменить, а после оператора SET нужно указать имя свойства.

Давайте посмотрим имена свойств которые нужно подставить вместо параметра имя\_свойства:

* SINGLE\_USER – перевести базу данных в однопользовательский режим. Только один пользователь сможет работать с базой;
* RESTRICTED\_USER – к базе данных разрешено подключаться только пользователям, которые принадлежат роли db\_owner, dbcreator или sysadmin;
* MULTI\_USER – нормальный многопользовательский режим, при котором действуют все права (используется по умолчанию);
* OFFLINE – отключить базу данных, подключения будут невозможны. Команды должна выполняться, когда к базе данных нет активных подключений. Вы при этом должны быть подключены к базе данных master.
* ONLINE – вернуть базу данных в активное состояние;
* READ\_ONLY - перевести базу данных в режим только для чтения, изменение данных будет невозможно;
* READ\_WRITE - вернуть базе данных полный доступ на запись и чтение;
* CURSOR\_CLOSE\_ON\_COMMIT ON – по завершении транзакции (принятии или откате) все открытые курсоры будут закрываться. Если ON заменить на OFF, то при нормальном завершении транзакции (принятии изменений) курсоры остаются открытыми. При откате все курсоры кроме INSENSITIVE и STATIC закрываются;
* RECOVERY FULL – использовать полную модель восстановления;
* BULK\_LOGGED – установить модель восстановления BULK\_LOGGED;
* SIMPLE – установить простую модель восстановления.

Некоторым опциям можно присвоить только значения ON или OFF, но для большинства из них предоставляется выбор из списка возможных значений. Каждый параметр базы данных имеет значение по умолчанию, которое устанавливается в базе данных model. Поэтому значения определенных опций по умолчанию можно модифицировать, изменив соответствующим образом базу данных model.

Все опции, значения которых можно изменять, можно разбить на несколько групп, наиболее важными из которых являются опции состояния, опции автоматических действий и опции SQL.

Опции состояния управляют следующими возможностями:

* доступом пользователей к базе данным (это опции single\_user, restricted\_user и multi\_user);
* статусом базы данных (это опции online, offline и emergency);
* режимом чтения и записи (опции read\_only и read\_write).

Опции автоматических операций управляют, среди прочего, остановом базы данных (опция auto\_close) и способом создания статистики индексов (опции auto\_create\_statistics и auto\_update\_statistics).

Опции SQL управляют соответствием базы данных и ее объектов стандарту ANSI. Значения всех операторов SQL можно узнать посредством функции DATABASE PROPERTY, а редактировать - с помощью инструкции ALTER DATABASE.

Опции восстановления full, bulk-logged и simple управляют процессом восстановления базы данных.

Теперь давайте посмотрим на примеры использования этих свойств:

Следующий пример разрешает доступ только одному пользователю:

ALTER DATABASE TestDatabase

SET SINGLE\_USER

Доступ только только пользователям ролей db\_owner, dbcreator или sysadmin:

ALTER DATABASE TestDatabase

SET RESTRICTED\_USER

Возвращаем нормальный многопользовательский режим:

ALTER DATABASE TestDatabase

SET MULTI\_USER

Вывести базу данных в off-line, т.е. доступ будет запрещен всем пользователям:

ALTER DATABASE TestDatabase

SET OFFLINE

Возобновить доступ к базе данных:

ALTER DATABASE TestDatabase

SET ONLINE

Перевести базу данных в режим только для чтения, любые изменения будут отклонены:

ALTER DATABASE TestDatabase

SET READ\_ONLY

Вернуть базе данных полный доступ на запись и чтение:

ALTER DATABASE TestDatabase

SET READ\_WRITE

По завершении транзакции (принятии или откате) все открытые курсоры будут закрываться:

ALTER DATABASE TestDatabase

SET CURSOR\_CLOSE\_ON\_COMMIT ON

Установить полную модель восстановления:

ALTER DATABASE TestDatabase

SET RECOVERY FULL

Установить модель восстановления BULK\_LOGGED:

ALTER DATABASE TestDatabase

SET BULK\_LOGGED

Установить простую модель восстановления:

ALTER DATABASE TestDatabase

SET SIMPLE

И последнее, что нам предстоит узнать – это возможность изменения раскладки (кодировки) по умолчанию для базы данных. Для этого выполняется команда:

ALTER DATABASE Имя\_базы

COLLATE имя\_кодировки

**Изменение таблиц**

Для модифицирования схемы таблицы применяется **инструкция ALTER TABLE**. Язык Transact-SQL позволяет осуществлять следующие виды изменений таблиц:

* добавлять и удалять столбцы;
* изменять свойства столбцов;
* добавлять и удалять ограничения для обеспечения целостности;
* разрешать или отключать ограничения;
* переименовывать таблицы и другие объекты базы данных.

**Синтаксис команды ALTER TABLE**

ALTER TABLE table

{ [ ALTER COLUMN column\_name

{ new\_data\_type [ ( precision [ , scale ] ) ]

[ COLLATE < collation\_name > ]

[ NULL | NOT NULL ]

| {ADD | DROP } ROWGUIDCOL }

]

| ADD

{ [ < column\_definition > ]

| column\_name AS computed\_column\_expression

} [ ,...n ]

| [ WITH CHECK | WITH NOCHECK ] ADD

{ < table\_constraint > } [ ,...n ]

| DROP

{ [ CONSTRAINT ] constraint\_name

| COLUMN column } [ ,...n ]

| { CHECK | NOCHECK } CONSTRAINT

{ ALL | constraint\_name [ ,...n ] }

| { ENABLE | DISABLE } TRIGGER

{ ALL | trigger\_name [ ,...n ] }

}

< column\_definition > ::=

{ column\_name data\_type }

[ [ DEFAULT constant\_expression ] [ WITH VALUES ]

| [ IDENTITY [ (seed , increment ) [ NOT FOR REPLICATION ] ] ]

]

[ ROWGUIDCOL ]

[ COLLATE < collation\_name > ]

[ < column\_constraint > ] [ ...n ]

< column\_constraint > ::=

[ CONSTRAINT constraint\_name ]

{ [ NULL | NOT NULL ]

| [ { PRIMARY KEY | UNIQUE }

[ CLUSTERED | NONCLUSTERED ]

[ WITH FILLFACTOR = fillfactor ]

[ ON { filegroup | DEFAULT } ]

]

| [ [ FOREIGN KEY ]

REFERENCES ref\_table [ ( ref\_column ) ]

[ ON DELETE { CASCADE | NO ACTION } ]

[ ON UPDATE { CASCADE | NO ACTION } ]

[ NOT FOR REPLICATION ]

]

| CHECK [ NOT FOR REPLICATION ]

( logical\_expression )

}

< table\_constraint > ::=

[ CONSTRAINT constraint\_name ]

{ [ { PRIMARY KEY | UNIQUE }

[ CLUSTERED | NONCLUSTERED ]

{ ( column [ ,...n ] ) }

[ WITH FILLFACTOR = fillfactor ]

[ ON {filegroup | DEFAULT } ]

]

| FOREIGN KEY

[ ( column [ ,...n ] ) ]

REFERENCES ref\_table [ ( ref\_column [ ,...n ] ) ]

[ ON DELETE { CASCADE | NO ACTION } ]

[ ON UPDATE { CASCADE | NO ACTION } ]

[ NOT FOR REPLICATION ]

| DEFAULT constant\_expression

[ FOR column ] [ WITH VALUES ]

| CHECK [ NOT FOR REPLICATION ]

( search\_conditions )

}

**Добавление и удаление столбцов**

Чтобы добавить новый столбец в существующую таблицу, в инструкции ALTER TABLE используется предложение **ADD**. В одной инструкции ALTER TABLE можно добавить только один столбец. Применение предложения ADD показано в примере ниже:

ALTER TABLE Employee

ADD PhoneNumber CHAR(12) NULL

В этом примере инструкция ALTER TABLE добавляет в таблицу Employee столбец PhoneNumber. Компонент Database Engine заполняет новый столбец значениями NULL или IDENTITY или указанными значениями по умолчанию. По этой причине новый столбец должен или поддерживать значения NULL, или для него должно быть указано значение по умолчанию.

Если попытаться добавить поле, не позволяющее нулевые значения и не указать значение по умолчанию, то поле не будет добавлено. Вместо этого вы увидите сообщение об ошибке:

*ALTER TABLE only allows columns to be added that can contain nulls or have a DEFAULT definition specified. Column 'vcFamil' cannot be added to table 'Peoples' because it does not allow nulls and does not specify a DEFAULT definition.*

*ALTER TABLE позволяет добавлять колонки только если они могут содержать нулевые значения, или имеют объявления значения по умолчанию (DEFAULT). Колонка "vcFamil" не может быть добавлена в таблицу Peoples, потому что не позволяет нулевых значений и не имеет значения по умолчанию.*

Почему такое ограничение? Дело в том, что если добавить в таблицу, уже содержащую строки новое поле, то в существующих строках будет установлено в новой колонке нулевое значение или значение по умолчанию. Если поле не поддерживает нулевых значений и не указать значение по умолчанию, то, что вставлять в новое поле для существующих строк? Нулевое значение нельзя, а какое другое значение указать, сервер не знает. Вот из-за этого конфликта и произошло такое ограничение.

Новый столбец нельзя вставить в таблицу в какой-либо конкретной позиции. Столбец, добавляемый предложением ADD, всегда вставляется в конец таблицы.

Столбцы из таблицы удаляются посредством предложения **DROP COLUMN**. Применение этого предложения показано в примере ниже:

ALTER TABLE Employee

DROP COLUMN PhoneNumber

В этом коде инструкция ALTER TABLE удаляет в таблице Employee столбец PhoneNumber, который был добавлен в эту таблицу предложением ADD ранее.

**Изменение свойств столбцов**

Для изменения свойств существующего столбца применяется предложение **ALTER COLUMN** инструкции ALTER TABLE. Изменению поддаются следующие свойства столбца:

* тип данных;
* поддержка значения NULL.

Применение предложения ALTER COLUMN показано в примере ниже:

ALTER TABLE Department

ALTER COLUMN Location NCHAR(25) NOT NULL

Инструкция ALTER TABLE в этом примере изменяет начальные свойства (nchar(40), значения NULL разрешены) столбца Location таблицы Department на новые (nchar(25), значения NULL не разрешены).

**Добавление и удаления ограничений для обеспечения целостности (ключей и проверок)**

Для добавления в таблицу новых ограничений для обеспечения целостности используется параметр **ADD CONSTRAINT** инструкции ALTER TABLE. В примере ниже показано использование параметра ADD CONSTRAINT для добавления проверочного ограничения и определения первичного ключа таблицы:

CREATE TABLE Sales (

Id INTEGER NOT NULL,

OrderDate DATE NOT NULL,

ShipDate DATE NOT NULL);

-- Добавляем ограничение для дат (поля OrderDate и ShipDate)

ALTER TABLE Sales

ADD CONSTRAINT order\_check CHECK(OrderDate <= ShipDate);

-- Добавляем первичный ключ для столбца Id

ALTER TABLE Sales

ADD CONSTRAINT pkey\_sales PRIMARY KEY(Id);

В этом примере сначала инструкцией CREATE TABLE создается таблица Sales, содержащая два столбца с типом данных DATE: OrderDate и ShipDate. Далее, инструкция ALTER TABLE определяет ограничение для обеспечения целостности order\_check, которое сравнивает значения обоих этих столбцов и выводит сообщение об ошибке, если дата отправки ShipDate более ранняя, чем дата заказа OrderDate. Далее инструкция ALTER TABLE используется для определения первичного ключа таблицы в столбце Id.

Ограничения для обеспечения целостности можно удалить посредством предложения **DROP CONSTRAINT** инструкции ALTER TABLE, как это показано в примере ниже:

ALTER TABLE Sales

DROP CONSTRAINT order\_check, pkey\_sales

Определения существующих ограничений нельзя модифицировать. Чтобы изменить ограничение, его сначала нужно удалить, а потом создать новое, содержащее требуемые модификации.

**Разрешение и запрещение ограничений**

Как упоминалось ранее, ограничение для обеспечения целостности всегда имеет имя, которое может быть объявленным или явно посредством опции CONSTRAINT, или неявно посредством системы. Имена всех ограничений таблицы (объявленных как явно, так и неявно) можно просмотреть с помощью системной процедуры **sp\_helpconstraint**.

В последующих операциях вставки или обновлений значений в соответствующий столбец ограничение по умолчанию обеспечивается принудительно. Кроме этого, при объявлении ограничения все существующие значения соответствующего столбца проверяются на удовлетворение условий ограничения. Начальная проверка не выполняется, если ограничение создается с параметром WITH NOCHECK. В таком случае ограничение будет проверяться только при последующих операциях вставки и обновлений значений соответствующего столбца. (Оба параметра - WITH CHECK и WITH NOCHECK - можно применять только с ограничениями проверки целостности CHECK и проверки внешнего ключа FOREIGN KEY.)

В примере ниже показано, как отключить все существующие ограничения таблицы:

ALTER TABLE Sales

NOCHECK CONSTRAINT ALL;

Все ограничения таблицы Sales отключаются посредством ключевого слова ALL. Применять опцию NOCHECK не рекомендуется, поскольку любые подавленные нарушения условий ограничения могут вызвать ошибки при будущих обновлениях.

**Переименование таблиц и других объектов баз данных**

Для изменения имени существующей таблицы (и любых других объектов базы данных, таких как база данных, представление или хранимая процедура) применяется системная процедура **sp\_rename**. В примере ниже показано использование этой системной процедуры:

-- Переименование таблицы Sales в BigSales

EXEC sp\_rename @objname = Sales, @newname = BigSales;

-- Переименование столбца OrderDate в таблице BigSales

EXEC sp\_rename @objname = 'BigSales.OrderDate', @newname = date\_order;

Использовать системную процедуру sp\_rename настоятельно не рекомендуется, поскольку изменение имен объектов может повлиять на другие объекты базы данных, которые ссылаются на них. Вместо этого следует удалить объект и воссоздать его с новым именем.

**Удаление объектов БД**

Все инструкции Transact-SQL для удаления объектов базы данных имеют следующий общий вид:

DROP тип\_объекта имя\_объекта

Для каждой инструкции CREATE object для создания объекта имеется соответствующая инструкция DROP object для удаления. Инструкция для удаления одной или нескольких баз данных имеет следующий вид:

DROP DATABASE database1 {, database2, ...}

Эта инструкция безвозвратно удаляет базу данных из системы баз данных. Для удаления одной или нескольких таблиц применяется следующая инструкция:

DROP TABLE table\_name1 {, table\_name2, ...}

При удалении таблицы удаляются все ее данные, индексы и триггеры. Но представления, созданные по удаленной таблице, не удаляются. Таблицу может удалить только пользователь, имеющий соответствующие разрешения.

Кроме объектов DATABASE и TABLE, в параметре objects инструкции DROP можно указывать, среди прочих, следующие объекты:

* TYPE;
* VIEW;
* SYNONYM;
* PROCEDURE;
* SCHEMA;
* INDEX;
* TRIGGER.

**Правила и объекты значений по умолчанию**

Объект по умолчанию и правило – это объекты, которые могут ограничивать одну или несколько полей или типы, определенные пользователем, делая возможным создавать их однажды и использовать регулярно.

Объект по умолчанию работает также как и ограничение, но только этот объект принадлежит базе данных, а не таблице. Если значение не указано, когда вы вставляете данные, для него будет использовано значение по умолчанию объекта, связанного с полем.

Объект значения по умолчанию создается следующим образом:

CREATE DEFAULT имя

AS выражение константы

После создания значения по умолчанию, вы должны связать его с колонкой или типом данных определенным пользователем с помощью вызова системной процедуры **sp\_bindefault**. Для отключения значения по умолчанию выполните системную процедуру **sp\_unbindefault**.

Следующий пример помещает шаблон номера телефона, если не указано реальное значение:

CREATE DEFAULT default\_phone\_no

AS '(000)00-00-00'

Следующая команда связывает созданный объект с полем "Phone" таблицы TestTable:

EXEC sp\_bindefault default\_ phone\_no, 'TestTable.Phone'

Объекты значений по умолчанию имеют свои ограничения:

* значения по умолчанию не применяются к уже существующим данным, а только к вновь введенным;
* можно использовать только простую логику с одной константой или функцией. Никакой логики здесь не может быть;
* тип данных значения должен соответствовать типу поля. Проверка этого произойдет только во время добавления новой строки, а не во время создания объекта, поэтому необходимо быть в этом моменте очень внимательным;
* у каждого поля может быть только одно значение по умолчанию. Это логично, ведь поле не может принять два значения. Если попытаться задать второе значение, то оно перезапишет уже существующее;
* не может быть обращения к другим таблицам на выборку данных;
* значение по умолчанию добавляется до проверок, поэтому если значение нарушает одно из ограничений, добавление записей станет невозможным;
* для удаления значения по умолчанию, его сначала нужно отвязать от объектов базы данных, к которым оно привязано.

Теперь переходим к правилам. Правила указывают доступные значения, которые вы можете вставить в колонку. Они гарантируют, что данные подпадают под определенный ряд значений, соответствуют определенному шаблону, или соответствуют определенному списку.

К столбцу, имеющему псевдоним типа данных, можно привязать только одно правило. Однако, кроме правила, со столбцом может быть связано одно или несколько ограничений CHECK. Если это так, соблюдаются все ограничивающие условия.

CREATE RULE [ schema\_name . ]   
 rule\_name AS condition\_expression   
 [ ; ]

schema\_name – имя схемы, к которой относится правило.  
rule\_name – имя нового правила. Имена правил должны соответствовать требованиям, предъявляемым к идентификаторам. Указывать имя владельца правила необязательно.

condition\_expression – условие или условия, определяющие правило. Правило может быть любым выражением, допустимым в предложении WHERE, и может включать такие элементы, как арифметические операторы, операторы отношений и предикаты (например, IN, LIKE и BETWEEN). Правило не может ссылаться на столбцы или другие объекты базы данных. В правило могут входить встроенные функции, не ссылающиеся на объекты базы данных. Определяемые пользователем функции использовать в правилах нельзя.

condition\_expression содержит одну переменную. Каждой локальной переменной предшествует знак @. Выражение соответствует значению, введенному при помощи инструкции UPDATE или INSERT. Для представления значения при создании правила можно использовать любое имя или символ, но первым знаком должен быть знак @.

После создания правила, вы должны связать его с колонкой или типом данных определенным пользователем с помощью вызова системной процедуры sp\_bindrule. Для отключения правила выполните системную процедуру sp\_unbinrule. В следующем примере, правило гарантирует, что поле "Pol" может содержать только букву М или Ж:

CREATE RULE rule\_pol

AS @Pol IN ('М', 'Ж')

Следующий пример связывает созданное правило с полем "Pol" таблицы TestTable:

EXEC sp\_bindrule rule\_pol, 'TestTable.Pol'

**Создание правила с диапазоном**

Следующий код создает правило, ограничивающее диапазон целых чисел, которые могут быть вставлены в столбец или столбцы, связанные с данным правилом.

CREATE RULE range\_rule

AS @range >= $1000

AND @range < $20000;

**Создание правила со списком**

Следующий код создает правило, ограничивающее значения, вставляемые в столбец или столбцы (к которым привязано данное правило) только теми значениями, которые указаны в правиле.

CREATE RULE list\_rule

AS @list IN ('1389','0736','0877');

**Создание правила с шаблоном**

Следующий код создает правило, позволяющее вставлять в столбец только значения, начинающиеся на два любых символа, за которыми следуют дефис (-), любое число символов или не следует никаких символов, и завершающиеся буквой из диапазона от a до z.

CREATE RULE supplier\_rule AS @value LIKE '\_\_-%[a-z]'

Для удаления значения по умолчанию и правила из базы данных используйте оператор DROP:

DROP DEFAULT имя [,…]

DROP RULE имя [,…]

Например, созданную ранее роль можно удалить командой:

DROP RULE rule\_pol

Правила являются достаточно мощным решением, но при этом они обладают достаточно большим количеством ограничений:

* новое правило не может повлиять на уже существующие значения, потому что оно срабатывает только при добавлении или изменении строк данных;
* в правилах можно использовать только не сложные вычисления с константами и функциями MS SQL Server;
* в отличии от ограничений, нельзя сравнивать значения полей;
* в правиле нельзя обращаться к таблицам для выборки данных. Чтобы была возможность для выборки, стоит обратить внимание на триггеры, которые мы будем рассматривать в 3-й главе;
* с одним столбцом можно связать только одно правило. Если бы можно было связывать два правила, то у нас появилась хоть какая-то возможность создавать более сложные запреты, но пока этого нет. Если попытаться связать правило со столбцом, у которого уже есть правило, то старое значение будет заменено новым;
* вполне логичное ограничение – тип данных, используемый в правиле должен совпадать с типом поля;
* связанное правило нельзя удалить. Необходимо сначала удалить связь.

Из-за такого большого количества ограничений для правил, их не очень и хочется использовать. Ограничения намного эффективнее и предоставляют нам больше возможностей. Мало того, корпорация Microsoft сама не рекомендует их использовать, и оставила эту возможность в SQL Server только для совместимости с предыдущими версиями баз данных.

## Вычисляемые столбцы

У таблицы в Microsoft SQL Server могут быть так называемые «*Вычисляемые столбцы*». Это такие столбцы, которые вычисляются на основе выражения с участием других столбцов этой же таблицы. Такие столбцы по умолчанию не хранятся физически, все значения обновляются, т.е. рассчитываются каждый раз при обращении к ним. Однако можно указать ключевое слово PERSISTED при создании подобного столбца, которое будет означать, что такой столбец нужно физически хранить, при этом значения данного столбца будет обновляться, когда будут вноситься любые изменения в столбцы, входящие в вычисляемое выражение.

К вычисляемым столбцам можно обращаться так же, как и к обычным столбцам в SQL запросах (*в списке выборки, в условии, при сортировке*).

Например, давайте создадим еще одну таблицу, в которой будет вычисляемый столбец.

Допустим, у нас таблица, в которой хранятся весовые товары.

CREATE TABLE TestTable3(

ProductId INT IDENTITY(1,1) NOT NULL,

ProductName VARCHAR(100) NOT NULL,

[Weight] DECIMAL(18, 2) NULL,

Price Money NULL,

Summa AS ([Weight] \* Price) PERSISTED

)

В данном примере, вычисляемый столбец Summa, он вычисляется на основе выражения, а именно перемножения веса на цену. При этом мы указали, что эти данные должны храниться физически, т.е. указали ключевое слово **PERSISTED**.

В случае если у Вас уже создана таблица, и Вам необходимо добавить в нее вычисляемый столбец, то это делается с помощью инструкции **ALTER TABLE**, и команды ADD. Для примера давайте добавим точно такой же вычисляемый столбец.

ALTER TABLE TestTable3 ADD [SummaDop] AS ([Weight] \* [Price]) PERSISTED

Добавление вычисляемых столбцов, как во время, так и после создания таблицы, доступно также и в графической среде SQL Server Management Studio в конструкторе таблиц. Для этого Вам нужно открыть конструктор таблиц, найти в нем раздел «*Спецификация вычисляемого столбца*» и в поле «*Формула*» указать соответствующую формулу.

**Задание:**

* 1. Запустить SQL Server Management Studio, выполнить подключение к серверу (используя параметры соединения, аналогичные практической 1).
  2. Отредактировать структуру центральной таблицы, добавив вычисляемое поле (по своему усмотрению).
  3. Создать правила и умолчания по заданию ниже (согласно варианту).

**В отчет предоставить:**

− текст SQL-запроса, выполняющий редактирование БД (добавление полей таблицы);

− скриншот диаграммы БД, показывающий добавленные поля (на скриншоте обязательно должно присутствовать имя (номер) компьютера);

− текст SQL-запроса, выполняющий создание необходимых правил и умолчаний, а также их привязку;

− скриншот структуры БД, показывающий добавленные правила и умолчания (на скриншоте обязательно должно присутствовать имя (номер) компьютера).

**Варианты:**

**1 подгруппа:**

**Вариант 1 Склад**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Адрес склада и со столбцом Название товара.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 0.

2.2. Связать умолчание со столбцом Количество.

**Вариант 2 Производство продуктов питания**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Название фирмы и со столбцом Название продукта питания.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 0.

2.2. Связать умолчание со столбцом Объем производства.

**Вариант 3 Учебная нагрузка**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Фамилия преподавателя и со столбцом Название дисциплины.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 1.

2.2. Связать умолчание со столбцом Количество часов в таблице УчебнаяНагрузка.

**Вариант 4 Растениеводство**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Название района и со столбцом Название культуры.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 0.

2.2. Связать умолчание со столбцом Урожайность.

**Вариант 5 Этнический состав стран**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Название страны и со столбцом Название национальности.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 100.

2.2. Связать умолчание со столбцом Численность народностей.

**Вариант 6 Курсы иностранных языков**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Фамилия учащегося и со столбцом Фамилия руководителя.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 1980.

2.2. Связать умолчание со столбцом Год рождения учащегося.

**Вариант 7 Услуги парикмахера**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Фамилия клиента и со столбцом Название услуги.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 500.

2.2. Связать умолчание со столбцом Цена товара.

**Вариант 8 Ателье по пошиву и ремонту одежды**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Название ателье и со столбцом Название услуги.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 0.

2.2. Связать умолчание со столбцом Стоимость.

**Вариант 9 Красная книга растений**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцов Название растения и со столбцом Название страны.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 0.

2.2. Связать умолчание со столбцом Количество корней за последние три года.

**Вариант 10 Животноводство**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Порода скота и со столбцом Название совхоза.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 1.

2.2. Связать умолчание со столбцом Поголовье.

**Вариант 11 Языки народов мира**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Название страны и со столбцом Название языка.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 0.

2.2. Связать умолчание со столбцом Численность.

**Вариант 12 Сеть компьютерных магазинов города**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Название комплектующей и со столбцом Название магазина.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 0.

2.2. Связать умолчание со столбцом Цена комплектующей.

**Вариант 13 Туристические фирмы города**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Название туристической фирмы города и со столбцом Название страны.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 26 000.

2.2. Связать умолчание со столбцом Стоимость тура.

**Вариант 14 Подписка на периодические издания организациями города**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Название организации города и со столбцом Название издания.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 50.

2.2. Связать умолчание со столбцом Цена издания.

**Вариант 15 Расписание занятий в вузе**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Название дисциплины и со столбцом Должность.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 32.

2.2. Связать умолчание со столбцом Количество часов.

**2 подгруппа:**

**Вариант 16 Справочная служба аптек**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Название аптеки и со столбцом Название лекарства.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 50.

2.2. Связать умолчание со столбцом Цена.

**Вариант 17 Сеть магазинов по продаже бытовой техники**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Адрес магазина и со столбцом Название техники.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 0.

2.2. Связать умолчание со столбцом Количество техники.

**Вариант 18 Стоматологическая поликлиника**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Фамилия пациента и со столбцом Название услуги.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 200.

2.2. Связать умолчание со столбцом Стоимость приёма.

**Вариант 19 Строительство жилья**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Наименование подрядчика и со столбцом Наименование заказчика.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 60.

2.2. Связать умолчание со столбцом Кол-во квартир.

**Вариант 20 Сотовые и интернет компании России**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Название сотового оператора и со столбцом Название города

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 1.

2.2. Связать умолчание со столбцом Количество абонентов.

**Вариант 21 Учет нарушений правил дорожного движения**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с латинской буквы.

1.2. Связать правило со столбцами Марка и Модель автомобиля.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 500.

2.2. Связать умолчание со столбцом Сумма штрафа нарушения.

**Вариант 22 Библиотека**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с руской буквы.

1.2. Связать правило со столбцом Фамилия читателя.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 1.

2.2. Связать умолчание со столбцом Кол-во экземпляров выдачи книги.

**Вариант 23 Прокат автомобилей**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с латинской буквы.

1.2. Связать правило со столбцами Марка и Модель автомобиля.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 1.

2.2. Связать умолчание со столбцом Кол-во дней проката.

**Вариант 24 Сотовые и интернет компании России**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Наименование услуги.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 10.

2.2. Связать умолчание со столбцом Процент вознаграждения исполнителя.

**Вариант 25 Приемная комиссия ВУЗа**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцами Факультет и Направление подготовки специальности.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание ‘Бюджет’.

2.2. Связать умолчание со столбцом Форма обучения на специальности.

**Вариант 26 Учет сделок с недвижимостью**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только цифры.

1.2. Связать правило со столбцом Кол-во комнат квартиры.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 10.

2.2. Связать умолчание со столбцом Площадь квартиры.

**Вариант 27 Потребительская корзина**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Наименование продукта.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 1.

2.2. Связать умолчание со столбцом Количество потребления.

**Вариант 28 Грузоперевозки**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с латинской буквы.

1.2. Связать правило со столбцами Марка и Модель автомобиля.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 1.

2.2. Связать умолчание со столбцом Кол-во в заявке.

**Вариант 29 Бухгалтерия**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только те значения, которые начинаются с русской буквы.

1.2. Связать правило со столбцом Должность работника.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 40.

2.2. Связать умолчание со столбцом Кол-во рабочих часов в табеле.

**Вариант 30 Учет договоров страхования**

1. Изучение оператора CREATE RULE для создания правил, использование системной хранимой процедуры sp\_bindrule для связывания правила с объектами БД.

1.1. Создать правило, которое позволяет использовать в качестве значений атрибута только цифры.

1.2. Связать правило со столбцом Телефон клиента.

2. Изучение оператора CREATE DEFAULT для создания умолчаний, использование системной хранимой процедуры sp\_bindefault для связывания умолчания с объектами БД.

2.1. Создать умолчание 0.

2.2. Связать умолчание со столбцом Процент скидки клиента.

**Контрольные вопросы:**

1. Изменения структуры каких объектов поддерживает T-SQL?
2. Какие действия по изменению свойств базы данных позволяет выполнять язык Transact-SQL?
3. Каким образом можно контролировать размер базы данных?
4. Опишите базовый синтаксис инструкции ALTER DATABASE.
5. Перечислите ситуации, которые увеличивают активность журнала транзакций.
6. Опишите способ переименования БД.
7. Перечислите свойства, которые можно задать, используя SET.
8. Какие виды изменений таблиц позволяет осуществлять язык Transact-SQL?
9. Опишите базовый синтаксис инструкции ALTER TABLE.
10. Что произойдет при добавлении поля, не позволяющего содержать нулевые значения и отсутствии значения по умолчанию?
11. Каким образом можно отключить все существующие ограничения таблицы?
12. С какими объектами может работать DROP?
13. Что такое умолчание? Как создать и связать с полем?
14. Какие ограничения имеет объект умолчание?
15. Что такое правило? Как создать и связать с полем?
16. Какие разновидности правил бывают?
17. Какие ограничения имеет объект правило?
18. Что такое вычисляемое поле?
19. Опишите процесс добавления вычисляемых полей при создании и после создания таблицы.
20. Можно ли обращаться к вычисляемым полям при сортировке? Ответ обоснуйте.

**Порядок оценивания:**

Оценка «**Отлично**» выставляется, если были выполнены все пункты задания, отчёт содержит всю необходимую информацию по проделанной работе, оформлен согласно требованиям, работа сдана и защищена в срок, в ходе защиты даны правильные ответы на 5 вопросов.

Оценка «**Хорошо**» выставляется, если были выполнены все пункты задания, отчёт содержит всю необходимую информацию по проделанной работе, оформлен согласно требованиям, работа сдана и защищена в срок, в ходе защиты даны правильные ответы на 4 вопроса.

Оценка «**Удовлетворительно**» выставляется, если были выполнены все пункты задания, отчёт содержит минимально необходимую информацию по проделанной работе, работа сдана и защищена с нарушением сроков, в ходе защиты даны правильные ответы на 2-3 вопроса.

Оценка «**Неудовлетворительно**» выставляется, если были выполнены не все пункты задания, отчёт содержит информацию, не позволяющую оценить ход выполнения работы, работа сдана с нарушением сроков, не защищена, либо в ходе защиты даны правильные ответы менее чем на 2 вопроса.

Работы, получившие оценку «Удовлетворительно» и выше являются засчитанными.

Работы, получившие оценку «Неудовлетворительно» являются обязательными к доработке и повторной сдаче. Итоговая оценка при повторной сдаче работы будет снижена на 1 балл.