Определение ограничений целостности

Таблицы с ограничениями в стандарте языка

При создании баз данных большое внимание должно быть уделено средствам поддержания данных в *целостном состоянии*. Рассмотрим предусмотренные стандартом языка *SQL* функции, которые предназначены для поддержания *целостности данных*. Эта *поддержка* включает средства задания ограничений, они вводятся с целью защиты *базы данных* от нарушения согласованности сохраняемых в ней данных. К таким типам поддержки *целостности данных* относятся:

- обязательные данные;
- ограничения для доменов полей;
- *целостность сущностей*;
- *ссылочная целостность* ;
- требования конкретного предприятия.

Большая часть перечисленных ограничений задается в операторах `CREATE TABLE` и `ALTER TABLE` .

Создание таблицы

В стандарте SQL дано несколько вариантов определения оператора *создания таблицы*, однако его базовый формат имеет следующий вид:

```
<определение_таблицы> ::=
CREATE TABLE имя_таблицы
{(имя_столбца тип_данных [ NOT NULL ][ UNIQUE]
[DEFAULT <значение>]
[ CHECK (<условие_выбора>)][,...n]}
[CONSTRAINT имя_ограничения]
[PRIMARY KEY (имя_столбца [,...n])
{[UNIQUE (имя_столбца [,...n])}
[FOREIGN KEY (имя_столбца_внешнего_ключа
    [,...n])
REFERENCES имя_род_таблицы
    [(имя_столбца_род_таблицы [,...n])],
[MATCH {PARTIAL | FULL}]
[ON UPDATE {CASCADE| SET NULL |SET DEFAULT
    |NO ACTION}]
[ON DELETE {CASCADE| SET NULL |SET DEFAULT
        |NO ACTION}]
{[CHECK(<условие_выбора>)][,...n]})
```

Ограничения

Представленная версия оператора *создания таблицы* включает средства определения требований *целостности данных*, а также другие конструкции. Имеются очень большие вариации в наборе функциональных возможностей этого оператора, реализованных в различных диалектах языка SQL. Рассмотрим назначение параметров команды, используемых для поддержания *целостности данных*.

Обязательные данные

Для некоторых столбцов требуется наличие в каждой строке таблицы конкретного и допустимого значения, отличного от опущенного значения или значения `NULL` . Для заданий ограничений подобного типа стандарт SQL предусматривает использование спецификации `NOT NULL` .

Требования конкретного предприятия

Обновления данных в таблицах могут быть ограничены существующими в организации требованиями (бизнес-правилами). Стандарт SQL позволяет реализовать бизнес-правила предприятий с помощью предложения `CHECK` и ключевого слова `UNIQUE` .

Ограничения для доменов полей

Каждый столбец имеет собственный домен - некоторый набор допустимых значений. Стандарт SQL предусматривает два различных механизма определения доменов. Первый состоит в использовании предложения `CHECK`, позволяющего задать требуемые ограничения для столбца или таблицы в целом, а второй предполагает применение оператора `CREATE DOMAIN`.

Целостность сущностей

Первичный ключ таблицы должен иметь уникальное непустое значение в каждой строке. Стандарт SQL позволяет задавать подобные требования поддержки *целостности данных* с помощью фразы `PRIMARY KEY`. В пределах таблицы она может указываться только один раз. Однако существует возможность гарантировать уникальность значений и для любых *альтернативных ключей* таблицы, что обеспечивает ключевое слово `UNIQUE`. Кроме того, при определении *альтернативных ключей* рекомендуется использовать и спецификаторы `NOT NULL`.

Ссылочная целостность

Внешние ключи представляют собой столбцы или наборы столбцов, предназначенные для связывания каждой из строк дочерней таблицы, содержащей этот *внешний ключ*, со строкой родительской таблицы, содержащей соответствующее значение *потенциального ключа*. Стандарт SQL предусматривает механизм определения *внешних ключей* с помощью предложения `FOREIGN KEY`, а фраза `REFERENCES` определяет имя родительской таблицы, т.е. таблицы, где находится соответствующий потенциальный ключ. При использовании этого предложения система отклонит выполнение любых операторов `INSERT` или `UPDATE`, с помощью которых будет предпринята попытка создать в дочерней таблице значение *внешнего ключа*, не соответствующее одному из уже существующих значений *потенциального ключа* родительской таблицы. Когда действия системы выполняются при поступлении операторов `UPDATE` и `DELETE`, содержащих попытку обновить или удалить значение *потенциального ключа* в родительской таблице, которому соответствует одна или более строк дочерней таблицы, то они зависят от *правил* поддержки *ссылочной целостности*, указанных во фразах `ON UPDATE` и `ON DELETE` предложения `FOREIGN KEY`. Если пользователь предпринимает попытку удалить из родительской таблицы строку, на которую ссылается одна или более строк дочерней таблицы, язык SQL предоставляет следующие возможности:

- `CASCADE` - выполняется удаление строки из родительской таблицы, сопровождающееся автоматическим удалением всех ссылающихся на нее строк дочерней таблицы;
- `SET NULL` - выполняется удаление строки из родительской таблицы, а во *внешние ключи* всех ссылающихся на нее строк дочерней таблицы записывается значение `NULL`;
- `SET DEFAULT` - выполняется удаление строки из родительской таблицы, а во *внешние ключи* всех ссылающихся на нее строк дочерней таблицы заносится значение, принимаемое по умолчанию;
- `NO ACTION` - операция удаления строки из родительской таблицы отменяется. Именно это значение используется по умолчанию в тех случаях, когда в описании *внешнего ключа* фраза `ON DELETE` опущена.

Те же самые *правила* применяются в языке SQL и тогда, когда значение *потенциального ключа* родительской таблицы обновляется.

Определитель `MATCH` позволяет уточнить способ обработки значения `NULL` во *внешнем ключе*.

При определении таблицы предложение `FOREIGN KEY` может указываться произвольное количество раз.

В операторе `CREATE TABLE` используется необязательная фраза `DEFAULT`, которая предназначена для задания принимаемого по умолчанию значения, когда в операторе `INSERT` значение в данном столбце будет отсутствовать.

Фраза `CONSTRAINT` позволяет задать имя ограничению, что позволит впоследствии отменить то или иное ограничение с помощью оператора `ALTER TABLE`.

Изменение и удаление таблицы

Для внесения изменений в уже созданные таблицы стандартом SQL предусмотрен оператор `ALTER TABLE`, предназначенный для выполнения следующих действий:

- добавление в таблицу нового столбца;
- удаление столбца из таблицы;
- добавление в определение таблицы нового ограничения;
- удаление из определения таблицы существующего ограничения;
- задание для столбца значения по умолчанию;
- отмена для столбца значения по умолчанию.

Оператор *изменения таблицы* имеет следующий обобщенный формат:

```
<изменение_таблицы> ::=
ALTER TABLE имя_таблицы
[ADD [COLUMN]имя_столбца тип_данных
        [ NOT NULL ][UNIQUE]
[DEFAULT <значение>][ CHECK (<условие_выбора>)]]
[DROP [COLUMN] имя_столбца [RESTRICT | CASCADE ]]
[ADD [CONSTRAINT [имя_ограничения]]
[{PRIMARY KEY (имя_столбца [,...n])
    |[UNIQUE (имя_столбца [,...n])}
|[FOREIGN KEY (имя_столбца_внешнего_ключа [,...n])
    REFERENCES имя_род_таблицы
       [(имя_столбца_род_таблицы [,...n])],
[ MATCH {PARTIAL | FULL}
    [ON UPDATE {CASCADE| SET NULL |
        SET DEFAULT | NO ACTION}]
    [ON DELETE {CASCADE| SET NULL |
        SET DEFAULT | NO ACTION}]
    |[CHECK(<условие_выбора>)][,...n]}]
[DROP CONSTRAINT имя_ограничения
     [RESTRICT | CASCADE]]
[ALTER [COLUMN] SET DEFAULT <значение>]
[ALTER [COLUMN] DROP DEFAULT]
```

Здесь параметры имеют то же самое назначение, что и в определении оператора `CREATE TABLE` .

Оператор `ALTER TABLE` реализован не во всех диалектах языка SQL. В некоторых диалектах он поддерживается, однако не позволяет удалять из таблицы уже существующие столбцы.

Для *удаления таблицы* используется команда `DROP TABLE` .

Таблицы в среде MS SQL Server

Создание таблицы

В процессе проектирования базы данных принимается решение о том, какие таблицы должны входить в базу данных, какие у них будут имена (идентификаторы), какие типы данных потребуются для построения таблиц и какие пользователи получат доступ к каждой из них. Кроме того, для эффективного *создания таблиц* необходимо ответить на следующие вопросы:

- Столбцы какого типа и размера будут составлять каждую из таблиц, какие требуется выбрать имена для столбцов таблиц?
- Какие столбцы могут содержать значение `NULL` ?
- Будут ли использованы *ограничения целостности*, значения по умолчанию и *правила* для столбцов?
- Необходимо ли индексирование столбцов, какие типы индексов будут применены для конкретных столбцов?
- Какие столбцы будут входить в *первичные* и *внешние ключи*.

Для *создания таблиц* в среде MS SQL Server используется команда:

```
<определение_таблицы> ::=
CREATE TABLE [ имя_базы_данных.[владелец].
             | владелец. ]имя_таблицы
             (<элемент_таблицы>[,...n])
```

где

```
<элемент_таблицы> ::=
{<определение_столбца>}
| <имя_столбца> AS <выражение>
| <ограничение_таблицы>
```

Обычно владельцем таблицы (dbo) является тот, кто ее создал.

<Выражение> задает значение для *вычисляемого столбца*. ***Вычисляемые столбцы*** - это виртуальные столбцы, т. е. физически в таблице они не хранятся и вычисляются с использованием значений столбцов той же таблицы. В выражении для *вычисляемого столбца* могут присутствовать имена обычных столбцов, константы и функции, связанные одним или несколькими операторами. Подзапросы в таком выражении участвовать не могут. *Вычисляемые столбцы* могут быть включены в раздел `SELECT` при указании списка столбцов, которые должны быть возвращены в результате выполнения запроса. *Вычисляемые столбцы* не могут входить во *внешний ключ*, для них не используются значения по умолчанию. Кроме того, *вычисляемые столбцы* не могут участвовать в операциях `INSERT` и `DELETE` .

```
<определение_столбца> ::=
{ имя_столбца <тип_данных>}
[ [ DEFAULT <выражение> ]
| [ IDENTITY (начало, шаг) [NOT FOR REPLICATION]]]]
[ROWGUIDCOL][<ограничение_столбца>][...n]]
```

В определении столбца обратим внимание на параметр `IDENTITY` , который указывает, что соответствующий столбец будет *столбцом-счетчиком*. Для таблицы может быть определен только один столбец с таким свойством. Можно дополнительно указать начальное значение и шаг приращения. Если эти значения не указываются, то по умолчанию они оба равны 1. Если с ключевым словом `IDENTITY` указано `NOT FOR REPLICATION` , то сервер не будет выполнять автоматического генерирования значений для этого столбца, а разрешит вставку в столбец произвольных значений.

В качестве ограничений используются *ограничения столбца* и *ограничения таблицы*. Различие между ними в том, что ***ограничение столбца*** применяется только к определенному полю, а ***ограничение таблицы*** - к группам из одного или более полей.

```
<ограничение_столбца>::=
[ CONSTRAINT имя_ограничения ]
{ [ NULL | NOT NULL ]
| [ {PRIMARY KEY | UNIQUE }
[ CLUSTERED | NONCLUSTERED ]
[ WITH FILLFACTOR=фактор_заполнения ]
[ ON {имя_группы_файлов | DEFAULT } ] ] ]
| [ [ FOREIGN KEY ]
REFERENCES имя_род_таблицы
    [(имя_столбца_род_таблицы) ]
[ ON DELETE { CASCADE | NO ACTION } ]
[ ON UPDATE { CASCADE | NO ACTION } ]
[ NOT FOR REPLICATION ]]
| CHECK [ NOT FOR REPLICATION](<лог_выражение>) }

<ограничение_таблицы>::=
[CONSTRAINT имя_ограничения ]
{ [ {PRIMARY KEY | UNIQUE }
    [ CLUSTERED | NONCLUSTERED ]
{(имя_столбца [ASC | DESC][,...n])}
[WITH FILLFACTOR=фактор_заполнения ]
[ON {имя_группы_файлов | DEFAULT } ]]
|FOREIGN KEY[(имя_столбца [,...n])]
REFERENCES имя_род_таблицы
    [(имя_столбца_род_таблицы [,...n])]
[ ON DELETE { CASCADE | NO ACTION } ]
[ ON UPDATE { CASCADE | NO ACTION } ]
| NOT  FOR REPLICATION ]
| CHECK [ NOT FOR REPLICATION ] (лог_выражение) }
```

Рассмотрим отдельные параметры представленных конструкций, связанные с ограничениями *целостности данных*. *Ограничения целостности* имеют приоритет над триггерами, *правилами* и значениями по умолчанию. К ***ограничениям целостности*** относятся *ограничение первичного ключа* `PRIMARY KEY`, *ограничение внешнего ключа* `FOREIGN KEY`, ограничение уникальности `UNIQUE`, ограничение значения `NULL`, ограничение на проверку `CHECK`.

Ограничение первичного ключа (PRIMARY KEY)

Таблица обычно имеет столбец или комбинацию столбцов, значения которых уникально идентифицируют каждую строку в таблице. Этот столбец (или столбцы) называется ***первичным ключом*** таблицы и нужен для обеспечения ее целостности. Если в *первичный ключ* входит более одного столбца, то значения в пределах одного столбца могут дублироваться, но любая комбинация значений всех столбцов *первичного ключа* должна быть уникальна.

При создании *первичного ключа* SQL Server автоматически создает уникальный индекс для столбцов, входящих в *первичный ключ*. Он ускоряет доступ к данным этих столбцов при использовании *первичного ключа* в запросах.

Таблица может иметь только одно ограничение `PRIMARY KEY`, причем ни один из включенных в *первичный ключ* столбцов не может принимать значение `NULL`. При попытке использовать в качестве *первичного ключа* столбец (или группу столбцов), для которого *ограничения первичного ключа* не выполняются, *первичный ключ* создан не будет, а система выдаст сообщение об ошибке.

Поскольку ограничение `PRIMARY KEY` гарантирует уникальность данных, оно часто определяется для *столбцов-счетчиков*. Создание *ограничения целостности* `PRIMARY KEY` возможно как при создании, так и при *изменении таблицы*. Одним из назначений *первичного ключа* является обеспечение *ссылочной целостности* данных нескольких таблиц. Естественно, это может быть реализовано только при определении соответствующих *внешних ключей* в других таблицах.

Ограничение внешнего ключа (FOREIGN KEY)

Ограничение внешнего ключа - это основной механизм для поддержания *ссылочной целостности* между таблицами реляционной базы данных. Столбец дочерней таблицы, определенный в качестве *внешнего ключа* в параметре `FOREIGN KEY`, применяется для ссылки на столбец родительской таблицы, являющийся в ней *первичным ключом*. Имя родительской таблицы и столбцы ее *первичного ключа* указываются в предложении `REFERENCES`. Данные в столбцах, определенных в качестве *внешнего ключа*, могут принимать только такие же значения, какие находятся в связанных с ним столбцах *первичного ключа* родительской таблицы. Совпадение имен столбцов для связи дочерней и родительской таблиц необязательно. *Первичный ключ* может быть определен для столбца с одним именем, в то время как столбец, на который наложено ограничение `FOREIGN KEY`, может иметь совершенно другое имя. Единственным требованием остается соответствие столбцов по типу и размеру данных.

На *первичный ключ* могут ссылаться не только столбцы других таблиц, но и столбцы, расположенные в той же таблице, что и собственно *первичный ключ*; это позволяет создавать рекурсивные структуры.

Внешний ключ может быть связан не только с *первичным ключом* другой таблицы. Он может быть определен и для столбцов с ограничением `UNIQUE` второй таблицы или любых других столбцов, но таблицы должны находиться в одной базе данных.

Столбцы *внешнего ключа* могут содержать значение `NULL`, однако проверка на ограничение `FOREIGN KEY` игнорируется. *Внешний ключ* может быть проиндексирован, тогда сервер будет быстрее отыскивать нужные данные. *Внешний ключ* определяется как при создании, так и при *изменении таблиц*.

Ограничение *ссылочной целостности* задает требование, согласно которому для каждой записи в дочерней таблице должна иметься запись в родительской таблице. При этом изменение значения столбца связи в записи родительской таблицы при наличии дочерней записи блокируется, равно как и удаление родительской записи (запрет каскадного изменения и удаления), что гарантируется параметрами `ON DELETE NO ACTION` и `ON UPDATE NO ACTION`, принятыми по умолчанию. Для разрешения каскадного воздействия следует использовать параметры `ON DELETE CASCADE` и `ON UPDATE CASCADE`.

Ограничение уникального ключа (UNIQUE)

Это ограничение задает требование уникальности значения поля (столбца) или группы полей (столбцов), входящих в *уникальный ключ*, по отношению к другим записям. Ограничение `UNIQUE` для столбца таблицы похоже на *первичный ключ*: для каждой строки данных в нем должны содержаться уникальные значения. Установив для некоторого столбца *ограничение первичного ключа*, можно одновременно установить для другого столбца ограничение `UNIQUE`. Отличие в *ограничении первичного* и *уникального ключа* заключается в том, что *первичный ключ* служит как для упорядочения данных в таблице, так и для соединения связанных между собой таблиц. Кроме того, при использовании ограничения `UNIQUE` допускается существование значения `NULL`, но лишь единственный раз.

Ограничение на значение (NOT NULL)

Для каждого столбца таблицы можно установить ограничение `NOT NULL`, запрещающее ввод в этот столбец нулевого значения.

Ограничение проверочное (CHECK) и правила

Данное ограничение используется для проверки допустимости данных, вводимых в конкретный столбец таблицы, т.е. ограничение `CHECK` обеспечивает еще один уровень защиты данных.

Ограничения целостности `CHECK` задают диапазон возможных значений для столбца или столбцов. В основе *ограничений целостности* `CHECK` лежит использование логических выражений.

Допускается применение нескольких ограничений `CHECK` к одному и тому же столбцу. В этом случае они будут применимы в той последовательности, в которой происходило их создание. Возможно применение одного и того же ограничения к разным столбцам и использование в логических выражениях значений других столбцов. Указание параметра `NOT FOR REPLICATION` предписывает не выполнять проверочных действий, если они выполняются подсистемой репликации.

Проверочные ограничения могут быть реализованы и с помощью *правил*. ***Правило*** представляет собой самостоятельный объект базы данных, который связывается со столбцом таблицы или пользовательским типом данных. Причем одно и то же *правило* может быть одновременно связано с несколькими столбцами и пользовательскими типами данных, что является неоспоримым преимуществом. Однако существенный недостаток заключается в том, что с каждым столбцом или типом данных может быть связано только одно *правило*. Разрешается комбинирование *ограничений целостности* `CHECK` с *правилами*. В этом случае выполняется проверка соответствия вводимого значения как *ограничениям целостности*, так и *правилам*.

Правило может быть создано командой:

```
CREATE RULE имя_правила AS выражение
```

Чтобы связать *правило* с тем или иным столбцом какой-либо таблицы, необходимо использовать системную хранимую процедуру:

```
sp_bindrule [@rulename=] 'rule'
[@objname=] 'object_name'
[,[@futureonly=['futureonly_flag']
```

Чтобы отменить *правила*, следует выполнить следующую процедуру:

```
sp_unbindrule [@objname=] 'object_name'
[,[@futureonly=['futureonly_flag']
```

Удаление *правила* производится командой

```
DROP RULE {имя_правила} [,...n].
```

Ограничение по умолчанию (DEFAULT)

Столбцу может быть присвоено значение по умолчанию. Оно будет актуальным в том случае, если пользователь не введет в столбец никакого иного значения.

Отдельно необходимо отметить пользу от использования значений по умолчанию при добавлении нового столбца в таблицу. Если для добавляемого столбца не разрешено хранение значений `NULL` и не определено значение по умолчанию, то операция добавления столбца закончится неудачей.

При определении в таблице столбца с параметром `ROWGUIDCOL` сервер автоматически определяет для него значение по умолчанию в виде функции `NEWID()`. Таким образом, для каждой новой строки будет автоматически генерироваться глобальный уникальный идентификатор.

Дополнительным механизмом использования значений по умолчанию являются объекты базы данных, созданные командой:

```
CREATE DEFAULT имя_умолчания AS константа
```

Умолчание связывается с тем или иным столбцом какой-либо таблицы с помощью процедуры:

```
sp_bindefault [@defname=] 'default',
[@objname=]  'object_name'
[,[@futureonly=] 'futureonly_flag'],
```

где

```
'object_name'
```

может быть представлен как

```
'имя_таблицы.имя_столбца'
```

Удаление *ограничения по умолчанию* выполняется командой

```
DROP DEFAULT {имя_умолчания} [,...n]
```

если предварительно это ограничение было удалено из всех таблиц процедурой

```
sp_unbindefault [@objname=] 'object_name'
[,[@futureonly=] 'futureonly_flag']
```

При *создании таблицы*, кроме рассмотренных приемов, можно указать необязательное ключевое слово `CONSTRAINT`, чтобы присвоить ограничению имя, уникальное в пределах базы данных.

Ключевые слова `CLUSTERED` и `NONCLUSTERED` позволяют создать для столбца *кластерный* или *некластерный индекс*. Для ограничения `PRIMARY KEY` по умолчанию создается *кластерный индекс*, а для ограничения `UNIQUE` - *некластерный*. В каждой таблице может быть создан лишь один *кластерный индекс*, отличительной особенностью которого является то, что в соответствии с ним изменяется физический порядок строк в таблице. `ASC` и `DESC` определяют метод упорядочения данных в индексе.

С помощью параметра `WITH FILLFACTOR=фактор_заполнения` задается степень заполнения индексных страниц при создании индекса. Значение фактора заполнения указывается в процентах и может изменяться в промежутке от 0 до 100.

Параметр `ON имя_группы_файлов` обозначает группу, в которой предполагается хранить таблицу.

Изменение таблицы

Изменения в таблицу можно внести командой:

```
<изменение_таблицы> ::=
ALTER TABLE имя_таблицы
{[ALTER COLUMN имя_столбца
{ тип_данных [(точность[,масштаб])]
[ NULL | NOT NULL ]
| {ADD | DROP } ROWGUIDCOL }]
| ADD { [<определение_столбца>]
| имя_столбца AS выражение } [,...n]
| [WITH CHECK | WITH NOCHECK ]
     ADD  { <ограничение-таблицы> } [,...n]
| DROP
{ [CONSTRAINT ] имя_ограничения
| COLUMN имя_столбца}[,...n]
| {CHECK | NOCHECK } CONSTRAINT
{ALL | имя_ограничения[,...n]}
| {ENABLE | DISABLE } TRIGGER
{ALL | имя_триггера [,...n]}}
```

В дополнение к уже названным параметрам определим параметр `{ENABLE | DISABLE } TRIGGER ALL`, предписывающий задействовать или отключить конкретный триггер или все триггера, связанные с таблицей.

Удаление таблицы

Удаление таблицы выполняется командой:

```
DROP TABLE имя_таблицы
```

Удалить можно любую таблицу, даже системную. К этому вопросу нужно подходить очень осторожно. Однако удалению не подлежат таблицы, если существуют объекты, ссылающиеся на них. К таким объектам относятся таблицы, связанные с удаляемой таблицей посредством *внешнего ключа*. Поэтому, прежде чем удалять родительскую таблицу, необходимо удалить либо *ограничение внешнего ключа* , либо дочерние таблицы. Если с таблицей связано хотя бы одно представление, то таблицу также удалить не удастся. Кроме того, связь с таблицей может быть установлена со стороны функций и процедур. Следовательно, перед *удалением таблицы* необходимо удалить все объекты базы данных, которые на нее ссылаются, либо изменить их таким образом, чтобы ссылок на удаляемую таблицу не было.

```
CREATE TABLE Товар
(КодТовара INT IDENTITY(1,1) PRIMARY KEY,
 Название  VARCHAR(50)       NOT NULL UNIQUE,
 Цена      MONEY             NOT NULL,
 Тип       VARCHAR(50)       NOT NULL,
 Сорт      VARCHAR(50)       NOT NULL
   CHECK(сорт in('первый','второй','третий')),
 Город     VARCHAR(50)       NOT NULL,
 Остаток   INT
   CHECK(остаток>=0))
```

Пример 9.1. Создание родительской таблицы Товар с ограничениями.

```
CREATE TABLE  Клиент
(КодКлиента INT IDENTITY(1,1) PRIMARY KEY,
 Фирма      VARCHAR(50)       NOT NULL,
 Фамилия    VARCHAR(50)       NOT NULL,
 Город      VARCHAR(50)       NOT NULL,
 Телефон    CHAR(10)          NOT NULL
   CHECK(Телефон LIKE
     '[1-9][0-9]-[0-9][0-9]-[0-9][0-9]'))
```

Пример 9.2. Создание родительской таблицы Клиент с ограничениями.

```
CREATE TABLE Сделка
(КодСделки  INT IDENTITY(1,1) PRIMARY KEY,
 КодТовара  INT       NOT NULL,
 КодКлиента INT       NOT NULL,
 Количество INT       NOT NULL DEFAULT 0,
 Дата       DATETIME  NOT NULL DEFAULT
                      GETDATE(),
CONSTRAINT  fk_Товар
FOREIGN KEY(КодТовара)  REFERENCES Товар,
CONSTRAINT  fk_Клиент
FOREIGN KEY(КодКлиента) REFERENCES Клиент)
```

Пример 9.3. Создание дочерней таблицы Сделка с ограничениями.

```
CREATE TABLE Склад
(КодТовара     INT PRIMARY KEY,
 Остаток       INT)
```

Пример 9.4. Создание таблицы Склад.

```
ALTER TABLE Сделка DROP CONSTRAINT fk_Товар
```

Пример 9.5. Удаление ограничения внешнего ключа.

```
ALTER TABLE Сделка ADD CONSTRAINT fk_Товар
FOREIGN KEY (КодТовара) REFERENCES товар
ON UPDATE NO ACTION ON DELETE NO ACTION
```

Пример 9.6. Добавление ограничения внешнего ключа, реализующего декларативную ссылочную целостность.

```
ALTER TABLE Сделка ADD CONSTRAINT fk_Товар
FOREIGN KEY (КодТовара) REFERENCES Товар
ON UPDATE CASCADE ON DELETE CASCADE
```

Пример 9.7. Добавления ограничения внешнего ключа, реализующего каскадные обновления и изменения.

```
ALTER TABLE Товар ADD Налог AS Цена*0.05
ALTER TABLE Товар DROP COLUMN Налог
```

Пример 9.8. Пример создания и удаления вычисляемого поля.

Пусть создана *таблица* без ограничений:

```
CREATE TABLE Товар
(КодТовара     INT,
 Название      VARCHAR(20),
 Тип           VARCHAR(20),
 Дата          DATETIME,
 Цена          MONEY,
 Остаток       INT)
```

Рассмотрим примеры внесения в таблицу всевозможных ограничений.

Пример 9.9. *Поле* `КодТовара` необходимо сделать первичным ключом. Выполнение следующей команды будет отвергнуто, поскольку *поле* КодТовара допускает внесение значений `NULL` .

```
ALTER TABLE Товар ADD CONSTRAINT pk1
PRIMARY KEY(КодТовара)
```

Пример 9.9. Поле КодТовара необходимо сделать первичным ключом.

Сначала нужно изменить объявление столбца `КодТовара` , запретив внесение значений `NULL` :

```
ALTER TABLE Товар
ALTER COLUMN КодТовара INT NOT NULL
```

И только потом создать *ограничение первичного ключа* :

```
ALTER TABLE Товар ADD CONSTRAINT pk1
PRIMARY KEY(КодТовара)
```

Пример 9.10. Удалить столбец целого типа и добавить столбец-счетчик.

```
ALTER TABLE Товар DROP  COLUMN КодТовара
ALTER TABLE Товар ADD
    КодТовара INT IDENTITY(1,1)
```

Пример 9.10. Удаление столбца целого типа и добавление столбца-счетчика.

Пример 9.11. Добавить *ограничение первичного ключа* .

```
ALTER TABLE Товар ADD CONSTRAINT pk1
PRIMARY KEY(КодТовара)
```

Пример 9.11. Добавление ограничений первичного ключа.

Пример 9.12. Изменить столбец, добавив ограничение `NOT NULL` .

```
ALTER TABLE Товар ALTER COLUMN
    Название VARCHAR(40) NOT NULL
```

Пример 9.12. Добавление ограничения NOT NULL.

Пример 9.13. Добавить ограничение уникальности значения.

```
ALTER TABLE Товар ADD CONSTRAINT
    u1 UNIQUE(Название)
```

Пример 9.13. Добавление ограничения уникальности значения.

Пример 9.14. Создать умолчание и добавить умолчание столбцу.

```
CREATE DEFAULT df1 AS 0
sp_bindefault 'df1', 'Товар.Остаток'

CREATE DEFAULT df2 AS GETDATE()
sp_bindefault 'df2', 'Товар.Дата'
```

Пример 9.14. Создание и добавление умолчания столбцу.

Пример 9.15. Создать правило и добавить правило столбцу.

```
CREATE RULE r1 AS @m IN
    ('мебель','бытовая химия','косметика')
sp_bindrule 'r1','Товар.Тип'
```

Пример 9.15. Создание и добавление правила столбцу.