Инструкции языка описания данных (Инструкции DDL)

В реляционной модели постулируется, что основными структурами данных являются домены и нормализованные «n-арные» отношения, т.е. таблицы.

Создание, изменение и уничтожение доменов

В стандартном SQL домен является именованным объектом схемы базы данных. Домены можно создавать (CREATE), изменять (ALTER) и ликвидировать (DROP). Имена доменов можно использовать при определении столбцов таблиц. Можно считать, что в SQL определение домена представляет собой вынесенное за пределы определения индивидуальной таблицы «родовое» определение столбца, которое можно использовать для определения различных реальных столбцов реальных базовых таблиц.

Определение домена

Для определения домена в SQL используется оператор CREATE DOMAIN. Общий синтаксис этого оператора следующий:

определение_домена ::= CREATE DOMAIN *имя_домена* [AS] *тип_данных*
[*значение_по_умолчанию*]
[*список_ограничений_целостности*]

Раздел *значение_по_умолчанию* имеет вид:

DEFAULT { *литерал* | *нульместная_функция* | NULL }

где нульместная_функция может задаваться в одной из следующих форм:

USER
CURRENT_USER
SESSION_USER
SYSTEM_USER
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP

Элемент списка *список_ограничений_целостности* имеет вид

[CONSTRAINT *имя_ограничения*] CHECK (*условное_выражение*)

Наиболее естественным (и наиболее распространенным) видом ограничения домена является следующий:

CHECK (VALUE IN (*список_допустимых_значений*))

Примеры определений доменов

```
CREATE DOMAIN EMP_NO AS INTEGER
    CHECK (VALUE BETWEEN 1 AND 10000);

CREATE DOMAIN SALARY AS NUMERIC (10, 2)
    DEFAULT 10000.00
    CHECK (VALUE BETWEEN 10000.00 AND 20000000.00)
    CONSTRAINT SAL_NOT_NULL CHECK (VALUE IS NOT NULL);
```

Замечание. Transact-SQL не поддерживает концепции домена. При попытке выполнить выше приведенные инструкции будет получено следующее сообщение:

```
Msg 343, Level 15, State 1, Line 1
Unknown object type 'DOMAIN' used in a CREATE, DROP, or ALTER statement.
```

Явные преобразования типов или доменов и оператор CAST

Неявные преобразования типов недостаточно гибки и иногда могут вызывать ошибки. Число допустимых неявных преобразований типов в SQL весьма ограничено. Однако в SQL существует специальный оператор CAST, с помощью которого можно явно преобразовывать типы или домены в более широких пределах допускаемых преобразований. Конструкция имеет следующий синтаксис:

CAST ({*скалярное выражение* | NULL } AS {*тип_данных* | *имя_домена*})

Примеры явного преобразования типа

A. Простое использование функций CAST

```
CAST([Цена] AS int)
```

B. Использование функции CAST с арифметическими операторами

В следующем примере вычисляется столбец значений путем деления суммарных продаж за год на проценты комиссионных . Результат преобразуется к типу данных int после округления до ближайшего целого числа.

```
CAST(ROUND([Суммарные продажи за год]/[ Проценты комиссионных], 0) AS int)
```

C. Использование функции CAST для сцепления строк

```
'The list price is ' + CAST(ListPrice AS varchar(12))
```

D. Использование функций CAST с типом данных datetime

```
CAST(GETDATE() AS nvarchar(30))
```

E. Использование функции CAST с временной переменной @myval

```
DECLARE @myval decimal (5, 2)
SET @myval = 193.57
SELECT CAST(CAST(@myval AS varbinary(20)) AS decimal(10,5))
```

Замечание. Transact-SQL поддерживает преобразование CAST.

Создание, изменение и уничтожение базовых таблиц

Базовая таблица SQL-ориентированной базы данных является прямым аналогом переменной отношения реляционной модели данных. Базовые (реально хранимые в базе данных) таблицы создаются (определяются) с использованием оператора CREATE TABLE. Для изменения определения базовой таблицы применяется оператор ALTER TABLE. Уничтожить хранимую таблицу (отменить ее определение) можно с помощью оператора DROP TABLE.

Определение базовой таблицы

Оператор создания базовой таблицы CREATE TABLE имеет следующий синтаксис:

определение_базовой_таблицы ::=
CREATE TABLE *имя_базовой_таблицы* (*список_элементов_базовой_таблицы)*

элемент_базовой_таблицы ::=
определение_столбца | *определение_ограничения_базовой_таблицы*

определение_столбца ::=
имя_столбца
{ *тип_данных* | *имя_домена* }
[*значение_по_умолчанию*]
[*список_определений_ограничений_столбца*]

значение_по_умолчанию ::=

DEFAULT { *литерал* | *нульместная_функция* | NULL }

определение_ограничения_столбца ::=
[CONSTRAINT *имя_ограничения*]
NOT NULL |
{ PRIMARY KEY | UNIQUE } |
ограничение_ссылочной_целостности |
CHECK (*условное_выражение*)

ограничение_ссылочной_целостности ::=
REFERENCES *имя_базовой_таблицы* [(*список_столбцов*)]
[ON DELETE *ссылочное_действие*]
[ON UPDATE *ссылочное_действие*]

Последняя синтаксическая конструкция работает и в случае определения внешнего ключа на уровне таблицы (в одном из определений табличных ограничений целостности). Поэтому обсуждение этого вопроса будет отложено до рассмотрения общего случая.

определение_ограничения_базовой_таблицы задается в следующем синтаксисе:

определение_ограничения_базовой_таблицы ::=
[CONSTRAINT *имя_ограничения*]
{ PRIMARY KEY | UNIQUE } (*список_столбцов*) |
FOREIGN KEY (*список_столбцов*) *ограничение_ссылочной_целостности* |
CHECK (*условное_выражение*)

Поддержка ссылочной целостности и ссылочные действия

В связи с определением ограничения внешнего ключа осталось рассмотреть еще два необязательных раздела – ON DELETE ссылочное_действие и ON UPDATE ссылочное_действие. Прежде всего, приведем синтаксическое правило:

ссылочное_действие ::=
{ NO ACTION | CASCADE | SET DEFAULT | SET NULL }

Чтобы объяснить, в каких случаях и каким образом выполняются эти действия, требуется сначала определить понятие ссылающейся строки (referencing row). Для данной строки t таблицы T строкой таблицы S, ссылающейся на строку t, называется каждая строка таблицы S, значение внешнего ключа которой совпадает со значением соответствующего потенциального ключа строки t.

Пусть определение ограничения внешнего ключа содержит раздел ON DELETE. Предположим, что предпринимается попытка удалить строку t из таблицы T. Тогда:

1) если в качестве требуемого ссылочного действия указано NO ACTION, то операция удаления отвергается, если ее выполнение вызвало бы нарушение ограничения внешнего ключа;
2) если в качестве требуемого ссылочного действия указано CASCADE, то строка t удаляется, и удаляются все строки, ссылающиеся на t;
3) если в качестве требуемого ссылочного действия указано SET DEFAULT, то строка t удаляется, и во всех столбцах, которые входят в состав внешнего ключа, всех строк, ссылающихся на строку t, проставляется заданное при их определении значение по умолчанию;
4) если в качестве требуемого ссылочного действия указано SET NULL, то строка t удаляется, и во всех столбцах, которые входят в состав внешнего ключа, всех строк, ссылающихся на строку t, проставляется NULL

Пусть определение ограничения внешнего ключа содержит раздел ON UPDATE. Предположим, что предпринимается попытка обновить столбцы соответствующего возможного ключа в строке t из таблицы T. Тогда:

- если в качестве требуемого ссылочного действия указано NO ACTION или RESTRICT, то операция обновления отвергается, если ее выполнение вызвало бы нарушение ограничения внешнего ключа;
- если в качестве требуемого ссылочного действия указано CASCADE, то строка t обновляется и соответствующим образом обновляются все строки, ссылающиеся на t (в них должным образом изменяются значения столбцов, входящих в состав внешнего ключа;

- если в качестве требуемого ссылочного действия указано SET DEFAULT, то строка t обновляется, и во всех столбцах, которые входят в состав внешнего ключа и соответствуют изменяемым столбцам таблицы T, всех строк, ссылающихся на строку t, проставляется заданное при их определении значение по умолчанию;
- если в качестве требуемого ссылочного действия указано SET NULL, то строка t обновляется, и во всех столбцах, которые входят в состав внешнего ключа и соответствуют изменяемым столбцам таблицы T, всех строк, ссылающихся на строку t, проставляется NULL.

Классификация ограничений

A. Ограничения NULL

Определяет, допустимы ли для столбца значения NULL. Параметр NULL не является ограничением в строгом смысле слова, но может быть указан так же, как и NOT NULL.

B. Ограничения PRIMARY KEY

1) В таблице возможно наличие только одного ограничения по первичному ключу.
2) Индекс, формируемый ограничением PRIMARY KEY, не может привести к выходу количества индексов в таблице за пределы в 999 некластеризованных индексов и 1 кластеризованный.
3) Если для ограничения PRIMARY KEY не указан параметр CLUSTERED или NONCLUSTERED, применяется параметр CLUSTERED, если для ограничения UNIQUE не определено кластеризованных индексов.
4) Все столбцы с ограничением PRIMARY KEY должны иметь признак NOT NULL. Если допустимость значения NULL не указана, то для всех столбцов с ограничением PRIMARY KEY устанавливается признак NOT NULL.
5) Если первичный ключ определен на столбце определяемого пользователем типа данных CLR, реализация этого типа должна поддерживать двоичную сортировку.

C. Ограничения UNIQUE

1) Если для ограничения UNIQUE не указан параметр CLUSTERED или NONCLUSTERED, по умолчанию применяется параметр NONCLUSTERED.
2) Каждое ограничение уникальности создает индекс. Количество ограничений UNIQUE не может привести к выходу количества индексов в таблице за пределы в 999 некластеризованных индексов и 1 кластеризованный.

D. Ограничения FOREIGN KEY

1) Если столбцу, имеющему ограничение внешнего ключа, задается значение, отличное от NULL, такое же значение должно существовать и в указываемом столбце; в противном случае будет возвращено сообщение о нарушении внешнего ключа.
2) Если не указаны исходные столбцы, ограничения FOREIGN KEY применяются к предшествующему столбцу.
3) Ограничения FOREIGN KEY могут ссылаться только на таблицы в пределах той же базы данных на том же сервере. Межбазовую ссылочную целостность необходимо реализовать посредством триггеров.
4) Ограничения FOREIGN KEY могут ссылаться на другие столбцы той же таблицы. Это называется самовызовом.
5) Предложение REFERENCES ограничения внешнего ключа на уровне столбца может содержать только один ссылочный столбец. Этот столбец должен принадлежать к тому же типу данных, что и столбец, для которого определяется ограничение.
6) Предложение REFERENCES ограничения внешнего ключа на уровне таблицы должно содержать такое же число ссылочных столбцов, какое содержится в списке столбцов в ограничении. Тип данных каждого ссылочного столбца должен также совпадать с типом соответствующего столбца в списке столбцов.
7) Если частью внешнего или ссылочного ключа является столбец типа **timestamp**, ключевые слова CASCADE, SET NULL и SET DEFAULT указывать нельзя.
8) Ключевые слова CASCADE, SET NULL, SET DEFAULT и NO ACTION можно сочетать в таблицах, имеющих взаимные ссылочные связи. Если компонент Компонент Database Engine обнаруживает ключевое слово NO ACTION, оно остановит и произведет откат связанных операций CASCADE, SET NULL и SET DEFAULT. Если инструкция DELETE содержит сочетание ключевых слов CASCADE, SET NULL, SET DEFAULT и NO ACTION, то все операции CASCADE, SET NULL и SET DEFAULT выполняются перед поиском компонентом Компонент Database Engine операции NO ACTION.
9) Компонент Компонент Database Engine не имеет стандартного предела на количество ограничений FOREIGN KEY, содержащихся в таблице, ссылающейся на другие таблицы, или на количество ограничений FOREIGN KEY в других таблицах, ссылающихся на указанную таблицу. Тем не менее фактическое количество ограничений FOREIGN KEY, доступных для использования, ограничивается конфигурацией оборудования, базы данных и приложения. Рекомендуется, чтобы таблица содержала не более 253 ограничений FOREIGN KEY, а также, чтобы на нее ссылалось не более 253 ограничений FOREIGN KEY. Предел эффективности в конкретном случае может более или менее зависеть от приложения и оборудования. При разработке базы данных и приложений следует учитывать стоимость принудительных ограничений FOREIGN KEY.
10) Ограничения FOREIGN KEY не применяются к временным таблицам.
11) Ограничения FOREIGN KEY могут ссылаться только на столбцы с ограничениями PRIMARY KEY или UNIQUE в таблице, на которую указывает ссылка, или на столбцы уникального индекса (UNIQUE INDEX) такой таблицы.

12) Если внешний ключ определен на столбце определяемого пользователем типа данных CLR, реализация этого типа должна поддерживать двоичную сортировку.
13) Столбцы, участвующие в связи по внешнему ключу, должны иметь одинаковую длину и масштаб.

E. Ограничения DEFAULT

1) Столбец может иметь только одно определение DEFAULT.
2) Ограничение DEFAULT может содержать значения констант, функции, функции без параметров SQL-92 или значение NULL.
3) Значение *константное_выражение* в определении DEFAULT не может ссылаться на другой столбец таблицы, а также на другие таблицы, представления или хранимые процедуры.
4) Определения DEFAULT нельзя создавать для столбцов с типом данных **timestamp** или столбцов со свойством IDENTITY.
5) Определения DEFAULT нельзя создавать для столбцов с псевдонимами типов данных, если такой тип привязан к определенному по умолчанию объекту.

F. Ограничения CHECK

1) Столбец может содержать любое количество ограничений CHECK, а условие может включать несколько логических выражений, соединенных операторами AND и OR. При указании нескольких ограничений CHECK для столбца их проверка производится в порядке создания.
2) Условие поиска должно возвращать логическое выражение и не может ссылаться на другую таблицу.
3) Ограничение CHECK уровня столбца может ссылаться только на ограничиваемый столбец, а ограничение CHECK уровня таблицы — только на столбцы этой таблицы.
4) Правила и ограничения CHECK выполняют одну и ту же функцию проверки данных при выполнении инструкций INSERT и UPDATE.
5) Если для столбца или столбцов задано правило либо одно или несколько ограничений CHECK, применяются все ограничения.
6) Ограничения CHECK нельзя определять для столбцов типов данных **text**, **ntext** или **image**.

Примеры определений базовых таблиц

```
CREATE TABLE EMPLOYEE
(
	ID SMALLINT NOT NULL,
	NAME VARCHAR(9),
	DEPT SMALLINT CHECK (DEPT BETWEEN 10 AND 100),
	JOB CHAR(5) CHECK (JOB IN ('Sales','Mgr','Clerk')),
	HIREDATE DATE,
	SALARY DECIMAL(7,2),
	COMM DECIMAL(7,2),
	PRIMARY KEY (ID),
	CONSTRAINT YEARSAL CHECK (YEAR(HIREDATE) > 1986 OR SALARY > 40500)
);

CREATE TABLE [dbo].[Shipping_Methods]
(
	[ShippingMethodID] [int] IDENTITY (1, 1) NOT NULL ,
	[ShippingMethod] [varchar] (25) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
);

CREATE TABLE DEPT
(
	DEPTNO SMALLINT NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 500, INCREMENT
BY 1),
	DEPTNAME VARCHAR (36) NOT NULL,
	MGRNO CHAR(6),
	ADMRDEPT SMALLINT NOT NULL,
	LOCATION CHAR(30)
);

create table Video
(
	VideoID char(5) constraint pk_Video primary key,
	MovID char(5) constraint fk_VideoMovie foreign key references Movie(MovID),
	Format char(4) not null,
```

```
      Price money not null
);
```

Изменение определения базовой таблицы

Оператор изменения определения базовой таблицы ALTER TABLE имеет следующий синтаксис:

изменение_базовой_таблицы ::=
ALTER TABLE *имя_базовой_аблицы* {
действие_по_изменению_определения_столбца
| *действие_по_изменению_определения_табличного_ограничения* }

Как видно из этого синтаксического правила, при выполнении одного оператора ALTER TABLE может быть выполнено либо действие по изменению определения столбца, либо действие по изменению определения табличного ограничения целостности.

Добавление, изменение или удаление определения столбца

Действие по изменению определения столбца специфицируется в следующем синтаксисе:

действие_по_изменению_определения_столбца ::=
ADD [COLUMN] *определение_столбца*
| ALTER [COLUMN] *имя_столбца* { SET *значение_по_умолчанию* | DROP DEFAULT }
| DROP [COLUMN] *имя_столбца* { RESTRICT | CASCADE }

Действие DROP COLUMN отменяет *определение* существующего столбца (удаляет его из таблицы). Действие DROP COLUMN отвергается, если:

(a) указанный столбец является единственным столбцом таблицы;

(b) или в этом действии присутствует спецификация RESTRICT, и данный столбец используется в определении каких-либо представлений или ограничений целостности.

Если в действии присутствует спецификация CASCADE, то его выполнение порождает неявное выполнение оператора DROP для всех представлений и ограничений целостности, в определении которых используется данный столбец.

Примеры изменения определения столбца

A. Добавление к таблице EMP нового столбца EMP_BONUS:

```
ALTER TABLE EMP
      ADD EMP_BONUS SALARY DEFAULT NULL
      CONSTRAINT BONSAL CHECK (VALUE < EMP_SAL);
```

B. При *определении столбца* EMP_SAL таблицы EMP для этого столбца явно не определялось *значение по умолчанию* (оно наследовалось из определения домена). Если в какой-то момент это стало неправильным (например, повысился размер минимальной зарплаты), можно установить новое *значение по умолчанию*:

```
ALTER TABLE EMP ALTER EMP_SAL SET DEFAULT 15000.00
```

C. При *определении столбца* DEPT_TOTAL_SAL таблицы DEPT для него было установлено *значение по умолчанию* 1000000. Можно отменить это *значение по умолчанию*:

```
ALTER TABLE DEPT ALTER DEPT_TOTAL_SAL DROP DEFAULT
```

D. Может оказаться разумным отменить *определение столбца* DEPT_EMP_NO, выполнив следующий оператор *ALTER TABLE*:

```
ALTER TABLE DEPT DROP DEPT_EMP_NO CASCADE
```

Спецификация CASCADE ведет к тому, что при выполнении оператора будет уничтожено не только *определение* указанного столбца, но и определения всех ограничений целостности и представлений, в которых используется уничтожаемый столбец. В нашем случае единственное связанное с этим столбцом ограничение целостности, определенное вне *определения столбца*, было бы отменено, даже если бы в операторе *отмены определения столбца* DEPT_EMP_NO содержалась спецификация RESTRICT, поскольку это единственное внешнее определение ограничения является ограничением только столбца DEPT_EMP_NO.

Изменение набора табличных ограничений

Действие по *изменению набора табличных ограничений* специфицируется в следующем синтаксисе:

*действие_по*_изменению_табличного_ограничения ::=
ADD [CONSTRAINT] *определение_ограничения_базовой таблицы* |
DROP CONSTRAINT *имя_ограничения* { RESTRICT | CASCADE }

Действие ADD [CONSTRAINT] позволяет добавить к набору существующих ограничений таблицы новое ограничение целостности. Можно считать, что новое ограничение добавляется через AND к конъюнкции существующих ограничений, как если бы оно определялось в составе оператора *CREATE TABLE*. Но здесь имеется одно существенное отличие. Если внимательно посмотреть на все возможные виды табличных ограничений, можно убедиться, что любое из них удовлетворяется на пустой таблице. Поэтому, какой бы набор табличных ограничений ни был определен при создании таблицы, это определение является допустимым и не препятствует выполнению оператора *CREATE TABLE*. При добавлении нового табличного ограничения с использованием действия ADD [CONSTRAINT] мы имеем другую ситуацию, поскольку таблица, скорее всего, уже содержит некоторый набор строк, для которого условное выражение нового ограничения может принять значение false. В этом случае выполнение оператора *ALTER TABLE*, включающего действие ADD [CONSTRAINT], отвергается.
Выполнение действия DROP CONSTRAINT приводит к отмене *определения* существующего табличного ограничения. Можно отменить определение только именованных табличных ограничений. Спецификации RESTRICT и CASCADE осмыслены только в том случае, если отменяемое ограничение является *ограничением потенциального ключа* (UNIQUE или PRIMARY KEY). При указании RESTRICT действие отвергается, если на данный возможный ключ ссылается хотя бы один внешний ключ. При указании CASCADE действие DROP CONSTRAINT выполняется в любом случае, и все определения таких внешних ключей также отменяются.

Примеры изменения набора табличных ограничений

A. Добавление к набору ограничений таблицы DEPT нового ограничения:

```
ALTER TABLE DEPT ADD CONSTRAINT TOTAL_INCOME
      CHECK (DEPT_TOTAL_SAL >= (
            SELECT SUM(EMP_SAL + COALESCE(EMP_BONUS,0))
            FROM EMP WHERE EMP.DEPT_NO = DEPT_NO))
```

Смысл этого ограничения следующий: суммарный доход служащих отдела не должен превышать объем зарплаты отдела.

B. В арифметическом выражении под знаком агрегатной операции SUM используется операция COALESCE. Эта двуместная операция определяется следующим образом:

```
COALESCE (x, y) IF x IS NOT NULL THEN x ELSE y
```

C. При определении таблицы EMP было специфицировано *проверочное табличное ограничение* PRO_EMP_NO, устанавливающее, что над одним проектом не должно работать более 50 служащих. Для отмены ограничения нужно выполнить следующий оператор:

```
ALTER TABLE EMP DROP CONSTRAINT PRO_EMP_NO
```

Отмена определения (уничтожение) базовой таблицы

Для отмены определения (уничтожения) *базовой таблицы* служит оператор *DROP TABLE*, задаваемый в следующем синтаксисе:

DROP TABLE *имя_базовой_таблицы* { RESTRICT | CADCADE }

При наличии спецификации RESTRICT выполнение оператора DROP TABLE отвергается, если имя таблицы используется в каком-либо определении представления или ограничения целостности. При наличии спецификации CASCADE оператор выполняется в любом случае, и все определения представлений и ограничений целостности, содержащие ссылки на данную таблицу, также отменяются.