Создание базы данных

Формы Бэкуса-Наура (BNF)

Язык, в терминах которого дается описание языка SQL, называется ***метаязыком***. При построении синтаксической диаграммы (дерева) главным образом используются условные обозначения, известные как стандартные формы Бэкуса-Наура (BNF), но, кроме того, введены некие дополнения, способствующие более глубокому пониманию смысла операций. Ниже приведен полный список обозначений.

Формы Бэкуса-Наура (BNF)

- Символ ::= означает равенство по определению. Используется для пояснения элементов синтаксической диаграммы оператора.

- Ключевые слова записываются прописными буквами. Они зарезервированы для обозначения и составляют часть оператора.

Формы Бэкуса-Наура (BNF)

- Необязательные элементы оператора заключены в квадратные скобки "[ ]".

[A] – повторение символа A 0 или 1 раз

- Вертикальная черта "|" указывает на то, что все предшествующие ей элементы являются необязательными и любой из них может быть заменен на другой, принадлежащий списку после этой черты.

Формы Бэкуса-Наура (BNF)

- Фигурные скобки "{ }" указывают, что все находящееся внутри них является единым целым при использовании других специальных символов (например, вертикальной черты или круглых скобок).

  {A} – повторение символа A произвольное число раз (включая 0 раз)

- Многоточие "..." (точнее три точки) означает, что предшествующая часть оператора может быть повторена любое число раз.

Формы Бэкуса-Наура (BNF)

- Многоточие, внутри которого есть запятая ".,.." (точнее, точка, запятая, две точки) указывает, что предшествующая часть оператора, состоящая из нескольких элементов, разделенных запятыми, может иметь произвольное число повторений. Запятую нельзя ставить после последнего элемента.

СОЗДАНИЕ БД

Этапы создания **БД**

1. создание* БД** (файл с расширением *.mdf***). В файле *БД* записываются сведения об основных объектах (*таблицах*, *индексах*, просмотрах и т.д.),

2. ***создание* журнала транзакций,** принадлежащего БД (файл с расширением ****.ldf***). Здесь записываются сведения о процессе работы с транзакциями (контроль целостности данных, состояния *БД* до и после выполнения транзакций).

В стандарте ANSI нет команды ***CREATE DATABASE.*** Но почти все платформы СУБД поддерживают какой-либо вариант этой команды.

При создании новой базы данных как образец используется база данных ***MODEL***.

Процедура создания базы данных обычно закрепляется только за администратором базы данных.

Определение базы данных

<определение_базы_данных> ::=

CREATE DATABASE имя_БД

[ON [PRIMARY]

[<определение_файла> [,...n]]

[,<определение_группы> [,...n]]]

[LOG ON {<определение_файла>[,...n] }]

Имя БД

Имя БД – стандартный идентификатор, допустимый в SQL. Если имя *БД* содержит пробелы или любые другие недопустимые символы, оно заключается в ограничители (квадратные скобки). Имя *БД* должно быть уникальным в пределах сервера и не может превышать 128 символов.

ON

ON – определяет список файлов на диске для размещения информации, хранящейся в *БД*.

Если в процессе использования *БД* планируется ее размещение на нескольких дисках, то можно создать так называемые ***вторичные*** *файлы БД* с расширением ****.ndf***. В этом случае основная информация о *БД* располагается в ***первичном*** (***PRIMARY***) файле, а при нехватке для него свободного места добавляемая информация будет размещаться во ***вторичном*** *файле*. Если первичный файл опущен, то основным является первый файл в списке. Основной файл содержит логическое начало базы данных. Подход, используемый в SQL-сервере, позволяет распределять содержимое *БД* по нескольким дисковым томам.

LOG ON

LOG ON – определяет список файлов на диске для размещения *журнала транзакций (ЖТ)*. Имя файла для *ЖТ* генерируется на основе имени *БД*, а в конце к нему добавляются символы ***log***.

Model.mdf –> Modellog.mdf

Определение файла

При создании *БД* можно определить набор файлов, из которых она будет состоять.

<определение_файла>::=
([NAME=логическое_имя_файла,]
FILENAME='физическое_имя_файла'
[,SIZE=размер_файла]
[,MAXSIZE={max_размер_файла |UNLIMITED
}] [, FILEGROWTH=величина_прироста]
)[,...n]

NAME=логическое_имя_файла

NAME=логическое_имя_файла – это имя файла, под которым он будет распознаваться при выполнении различных SQL-команд.

FILENAME='физическое_имя_файла' – это имя файла, который будет создан на жестком диске. Это имя останется за файлом на уровне операционной системы.

SIZE=размер_файла

SIZE=размер_файла определяет первоначальный размер файла; минимальный размер параметра – ***512 Кб;*** если он не указан, то по умолчанию принимается ***1 Мб***.

MAXSIZE={max_размер_файла} определяет максимальный размер файла базы данных. При задании параметра ***UNLIMITED*** максимальный размер базы данных ограничивается свободным местом на диске.

FILEGROWTH=величина_прироста

FILEGROWTH=величина_прироста – величина автоматического прироста размера базы данных. Приращение – это либо абсолютная величина в мегабайтах либо процентное соотношение.

Если ***FILEGROWTH*** не задан, то файл за одно увеличение будет увеличиваться на 10 % (но не менее, чем на 64 Кб.)

<определение_группы>::=

Дополнительные файлы могут быть включены в группу:

<определение_группы>::=

FILEGROUP имя_группы_файлов
<определение_файла>[,...]

Пример 1. Создать *БД*, при этом для данных определить 3 файла на дисках ***D***, ***E***, ***F***, для *журнала транзакций* – 2 файла на дисках ***H*** и ***M***.

```
CREATE DATABASE Archiv
ON PRIMARY
( NAME=Arch1,   FILENAME='d:\user\data\archdat1.mdf',
SIZE=100MB, MAXSIZE=200, FILEGROWTH=20),
(NAME=Arch2,   FILENAME='e:\user\data\archdat2.mdf',
SIZE=100MB, MAXSIZE=200, FILEGROWTH=20),
(NAME=Arch3,     FILENAME='f:\user\data\archdat3.mdf',
SIZE=100MB, MAXSIZE=200, FILEGROWTH=20),
LOG ON  (NAME=Archlog1,
FILENAME='h:\user\data\archlog1.ldf',      SIZE=100MB,
MAXSIZE=200, FILEGROWTH=20),
 (NAME=Archlog2,      FILENAME='m:\user\data\archlog2.ldf',
SIZE=100MB, MAXSIZE=200, FILEGROWTH=20);
```

Краткая форма оператора создания базы данных

CREATE DATABASE имя_базы_данных;

В этом случае все значения параметров задаются по умолчанию.

Пример 2. Создать базу данных Magasin с параметрами по умолчанию:

CREATE DATABASE Magasin;

Изменение БД

<изменение_базы_данных> ::=
ALTER DATABASE имя_базы_данных
{ ADD FILE <определение_файла>[,...n]
[TO FILEGROUP имя_группы_файлов]
| ADD LOG FILE <определение_файла>[,...n]
| REMOVE FILE логическое_имя_файла
| ADD FILEGROUP имя_группы_файлов
| REMOVE FILEGROUP имя_группы_файлов
| MODIFY NAME = new_database_name
| MODIFY FILEGROUP имя_группы_файлов
<свойства_группы_файлов>};

Удаление БД

Удаление *БД* осуществляется командой:

DROP DATABASE имя_базы_данных [,...n]

Удаляются все содержащиеся в *БД* объекты, освобождает распределенное под нее место на диске и уничтожает все ссылки на нее из БД ***master***. Для исполнения операции удаления *БД* пользователь должен обладать соответствующими правами.

Создание таблицы

Приступая к *созданию таблицы*, необходимо ответить на ряд вопросов:

- Как будет называться *таблица*?
- Как будут называться *столбцы* (поля) *таблицы*?
- Какие типы данных будут закреплены за каждым *столбцом*?
- Какой размер памяти должен быть выделен для хранения каждого *столбца*?
- Какие *столбцы таблицы* требуют обязательного ввода?
- Из каких *столбцов* будет состоять РК?

Упрощённый синтаксис этой команды

CREATE TABLE <имя таблицы>

({<имя столбца> <тип данных>
[(<размер>)]

[<ограничения целостности поля>…]} .,..

[, <ограничения целостности таблицы>.,..]
);

Базовый синтаксис команды создания таблицы

<определение_таблицы> ::=
CREATE TABLE имя_таблицы
({имя_столбца тип_данных [NOT NULL]
[[PRIMARY KEY | UNIQUE] [IDENTITY [n,m]]
[DEFAULT <значение>]
[IDENTITY [(стартовое_значение, инкремент)]]
[FOREIGN KEY REFERENCES имя_род_таблицы
[(имя_столбца_род_таблицы)]
[CHECK (<условие_выбора>)] [,...n]
[ON UPDATE {CASCADE | NO ACTION }]
[ON DELETE {CASCADE | NO ACTION }]});

имя_столбца тип_данных

имя_столбца – идентификатор столбца

тип_данных - тип данных столбца.

Обязательно должен быть указан размер для типов ***CHAR*** или ***VARCHAR*** .

NOT NULL

[NOT] NULL – ***NULL*** используется для указания того, что в данном *столбце* могут содержаться значения ***NULL,*** т. е. данные недоступны, опущены или недопустимы.

Если указано ключевое слово ***NOT NULL***, то будут отклонены любые попытки поместить значение ***NULL*** в данный столбец.

UNIQUE

UNIQUE – уникальное значение поля в пределах столбца таблицы.

PRIMARY KEY

- Создает первичный ключ таблицы. Каждая таблица может иметь только один первичный ключ.
- Обеспечивает отсутствие повторяющихся столбцов.
- Не допускает неопределенных значений ни в одной компоненте первичного ключа.

Таким образом, ***PRIMARY KEY*** является комбинацией ***NOT NULL*** и ***UNIQUE***

IDENTITY [n, m]

Для колонки с таким свойством сервером автоматически генерируется возрастающая последовательность, начиная с ***n*** и приращением ***m***. Если какой-либо параметр опущен, то по умолчанию принимается единица.

Внимание: Сервер не гарантирует непрерывность значений - в реальных данных таблицы могут появляться разрывы.

DEFAULT

[DEFAULT <значение>] - значение по умолчанию.

Так, при добавлении новой записи столбец с таким ограничением автоматически получит указанное значение.

FOREIGN KEY

[FOREIGN KEY REFERENCES имя_род_таблицы
[(имя_столбца_род_таблицы)]

1. Задает столбец или набор столбцов в качестве внешнего ключа таблицы.

2. Внешний ключ (FK) используется для поддержания ссылочной целостности. Он устанавливает связь с первичным или уникальным ключом в той же самой таблице или между таблицами.

3. Данные в столбце, определенном как ***FK***, могут принимать только ***такие же*** значения, какие находятся в связанных с ним столбцах ***первичного ключа*** родительской таблицы.

4. Значение внешнего ключа должно совпадать с существующим значением в родительской таблице или быть неопределенным (**NULL**).

FOREIGN KEY

Требование – соответствие столбцов ***РК*** и ***FK*** по типу и размеру данных

- если FK ссылается на РК другого отношения, имена полей РК можно не указывать;
- если FK составной, список полей, входящих в ключ, указывается после перечисления всех полей таблицы с ключевым словом ***FOREIGN KEY.***

CHECK

[CHECK (<условие_выбора>)] [,...n]

используется для проверки допустимости данных, вводимых в конкретный столбец таблицы. Это еще один уровень защиты данных.

CHECK задает диапазон возможных значений для столбца или столбцов. Допускается применение нескольких ограничений ***CHECK*** к одному и тому же столбцу. 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.

Основные типы ***условий выбора***:

1. ***Сравнение***
2. ***Диапазон***
3. ***Принадлежность множеству***
4. ***Соответствие шаблону***
5. ***Отсутствие значений (Значение NULL)***

1. Сравнение

Сравниваются результаты вычисления одного выражения с результатами вычисления другого.

Условие поиска при сравнении имеет вид

выр1 θ выр2,

где **θ** одна из операций сравнения.

В языке SQL можно использовать следующие ***операторы сравнения***:

- = – равенство;
- < – меньше;
- > – больше;
- <= – меньше или равно;
- >= – больше или равно;
- <> (!=) – не равно.

Пример 3. ***Цена*** INT NOT NULL CHECK (***Цена*** > 100 AND ***Цена*** < 200)

2. Диапазон

Проверяется, попадает ли результат вычисления выражения в заданный *диапазон* значений.

Диапазон значений задается с помощью конструкции

выражение BETWEEN нижняя граница AND верхняя граница

Диапазон определяется своими минимальным и максимальным значениями. При этом указанные значения включаются в *условие поиска*. При использовании отрицания ***NOT BETWEEN*** требуется, чтобы проверяемое значение лежало вне границ заданного *диапазона*.

Пример 4. ***Цена*** INT NOT NULL CHECK (***Цена*** BETWEEN 100 AND 200)

3. Принадлежность множеству

проверяется, принадлежит ли результат вычислений выражения заданному множеству значений.

Вхождение во множество задается с помощью конструкции

[NOT] IN (значение [, значение…])

Оператор ***IN*** используется для *сравнения* некоторого значения со списком заданных значений, при этом проверяется, соответствует ли результат вычисления выражения одному из значений в предоставленном списке. При помощи оператора ***IN*** может быть достигнут тот же результат, что и в случае применения оператора ***OR***, однако оператор ***IN*** выполняется быстрее.

3. Принадлежность множеству

NOT IN используется для отбора любых значений, кроме тех, которые указаны в представленном списке.

Пример 5.

Copm VARCHAR(6) NOT NULL

CHECK (***Copm IN*** (‘Первый',‘Второй',‘Третий'))

4. Соответствие шаблону

проверяется, отвечает ли некоторое строковое значение заданному шаблону.

С помощью оператора LIKE можно выполнять* сравнение *выражения с заданным шаблоном.

Символы-заменители, используемые в шаблоне:

% – вместо этого символа может быть подставлено любое количество произвольных символов.

_ заменяет один символ строки.

[] – вместо символа строки будет подставлен один из возможных символов, указанный в этих ограничителях.

[^] – вместо соответствующего символа строки будут подставлены все символы, кроме указанных в ограничителях.

4. Соответствие шаблону

Пример 6.

Телефон CHAR(10) NOT NULL

CHECK(Телефон LIKE '[1-9][0-9]-[0-9][0-9]-[0-9][0-9]')

Пример 7.

Город CHAR(15) NOT NULL CHECK (Город **LIKE** 'М%‘)

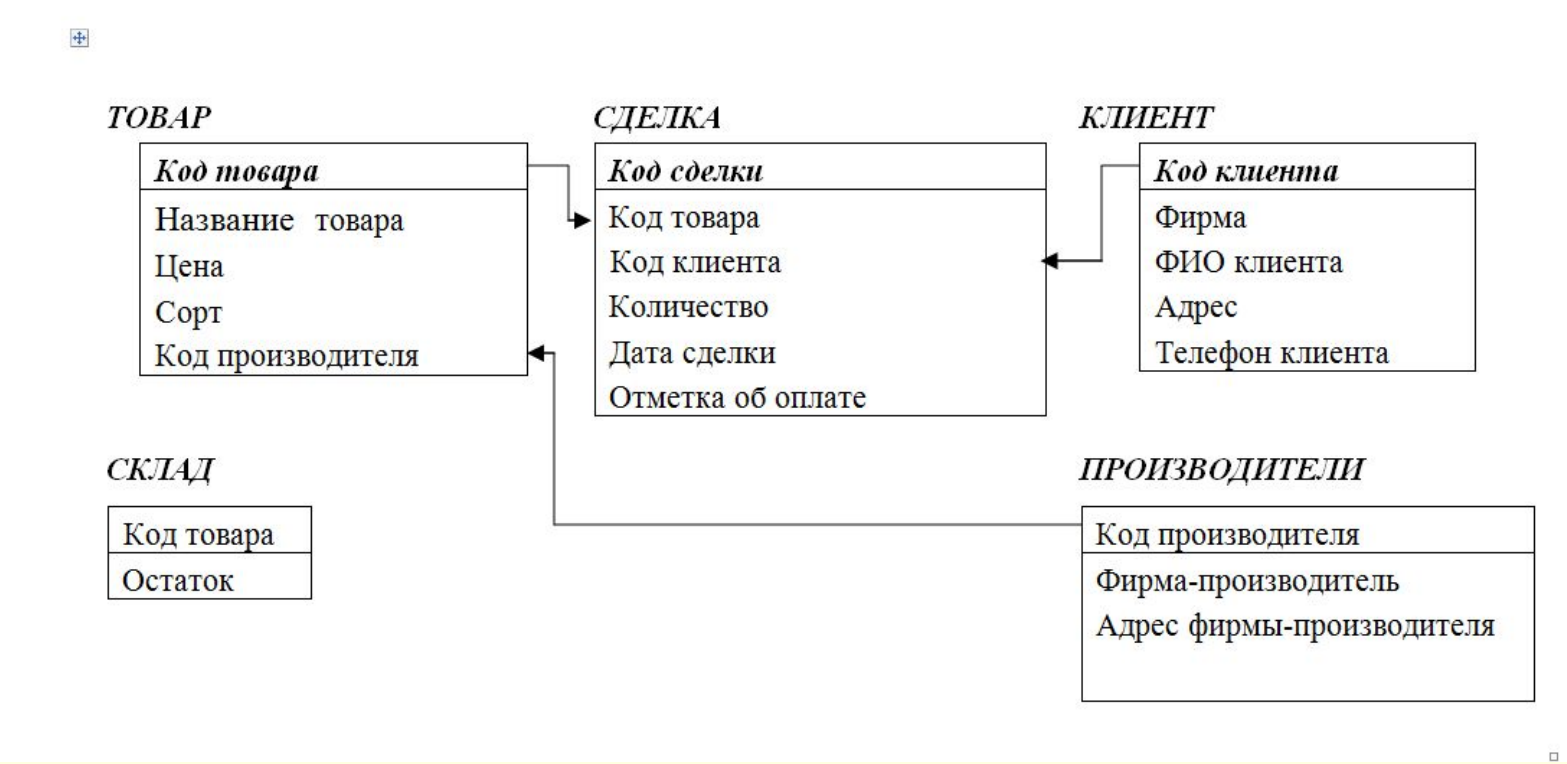
первый символ "М", далее любое количество любых символов, например, ***М****осква,* ***Минск***.

5. Отсутствие значений (Значение NULL)

Допускается наличие неопределенных значений в столбце.

Задается на уровне столбца.

База данных Magasin



Пример 7. Создание таблицы Сделка

```
CREATE TABLE Сделка
( КодСделки  INT IDENTITY (1,1)  PRIMARY KEY,
  КодТовара  INT   NOT NULL FOREIGN KEY  REFERENCES Товар,
  КодКлиента INT    NOT NULL  FOREIGN KEY  REFERENCES
    Клиент,
  Количество INT   NOT NULL DEFAULT 0  CHECK
    (Количество<200),
  Дата    DATETIME  NOT NULL DEFAULT    GETDATE(),
  КодПроизводителя  INT   NOT NULL,
  ОтметкаОбОплате CHAR (3)  CHECK (ОтметкаОбОплате IN
    ('да', 'нет')),
  CONSTRAINT  fk_Производитель FOREIGN KEY
    (КодПроизводителя) REFERENCES Производитель)
```

Пример 8. Создание таблицы Клиент

```
CREATE TABLE  Клиент
(КодКлиента INT IDENTITY(1,1) PRIMARY KEY,
Фирма   VARCHAR(50)   NOT NULL,
Фамилия    VARCHAR(50)       NOT NULL,
Адрес      VARCHAR(50)       NOT NULL,
Телефон    CHAR(8)           NOT NULL
CHECK(Телефон LIKE
'[1-9][0-9]-[0-9][0-9]-[0-9][0-9]'))
```

ИЗМЕНЕНИЕ ТАБЛИЦЫ

Изменение таблицы

Структура существующей *таблицы* может быть модифицирована с помощью команды ***ALTER TABLE,*** упрощенный синтаксис которой представлен ниже:

```
ALTER TABLE  имя_таблицы
{[ALTER COLUMN имя_столбца
{новый_тип_данных [ NULL | NOT NULL ]}]
 | ADD  { [имя_столбца тип_данных]  |
имя_столбца AS выражение } [,...n]|
DROP COLUMN {имя_столбца}[,...n]  }
```

Изменение таблицы

Некоторые реализации могут ограничить разработчика в использовании некоторых опций команды ***ALTER TABLE***.

Например, может оказаться недопустимым удаление *столбцов* из существующей *таблицы*. Чтобы добиться этого, сначала потребуется удалить саму *таблицу* и только потом заново ее построить с нужными *столбцами*. Причем уже внесенные в *таблицу* данные будут потеряны.

Изменение таблицы

Возможны трудности, связанные с удалением из *таблицы столбца*, который зависит от некоторого *столбца* другой *таблицы*. В таком случае сначала придется удалить ограничение *столбца*, а затем сам *столбец*.

Пример 9

1. Добавить в таблицу Клиент столбец ПаспДанные

ALTER TABLE Клиент ADD ПаспДанные varchar (10)

2. Изменить тип столбца ПаспДанные

ALTER TABLE Клиент ALTER COLUMN ПаспДанные int

3. Удалить из таблицы Клиент столбец ПаспДанные

ALTER TABLE Клиент DROP COLUMN ПаспДанные

УДАЛЕНИЕ ТАБЛИЦЫ

DROP TABLE

С течением времени структура *БД* меняется: создаются новые *таблицы*, а прежние становятся ненужными и удаляются из *БД* с помощью оператора:

DROP TABLE имя_таблицы [RESTRICT | CASCADE]

RESTRICT

Если указано ключевое слово ***RESTRICT***, то при наличии в *БД* хотя бы одного объекта, существование которого зависит от удаляемой *таблицы*, выполнение оператора ***DROP TABLE*** будет отменено.

CASCADE

Если указано ключевое слово ***CASCADE***, автоматически удаляются и все прочие объекты *БД*, чье существование зависит от удаляемой *таблицы*, а также другие объекты, зависящие от удаляемых объектов.

Общий эффект от выполнения оператора ***DROP TABLE*** с ключевым словом ***CASCADE*** может оказаться весьма ощутимым, поэтому подобные операторы следует использовать с максимальной осторожностью.

DROP TABLE

Чаще всего оператор ***DROP TABLE*** используется для исправления ошибок, допущенных при *создании таблицы*.

Если *таблица* была создана с некорректной структурой, можно воспользоваться оператором ***DROP TABLE*** для ее *удаления*, после чего создать *таблицу* заново.

TRUNCATE TABLE

Команда ***DROP TABLE*** удалит не только указанную *таблицу*, но и все входящие в нее *строки* данных.

Если требуется удалить из *таблицы* лишь данные, сохранив структуру *таблицы*, следует воспользоваться командой

TRUNCATE TABLE имя_таблицы

Эта команда делает то же самое, что и ***DELETE FROM***, но быстрее

Операторы модификации данных

ОПЕРАТОР ДОБАВЛЕНИЯ

Оператор добавления *INSERT INTO*

Формат оператора:

INSERT INTO <имя_таблицы> [(имя_столбца [,...n])] VALUES (значение [,...n])

Эта форма оператора ***INSERT*** с параметром ***VALUES*** предназначена для вставки единственной строки в указанную таблицу.

INSERT INTO

Список столбцов указывает столбцы, которым будут присвоены значения в добавляемых записях.

Список может быть опущен. Тогда подразумеваются все столбцы таблицы (кроме объявленных как счетчик), причем в порядке, установленном при создании таблицы.

Если в операторе ***INSERT*** указывается конкретный список имен полей, то любые пропущенные в нем столбцы должны быть объявлены при создании таблицы как допускающие значение ***NULL***, за исключением тех случаев, когда при описании столбца использовался параметр ***DEFAULT***.

Список значений должен соответствовать списку столбцов следующим образом:

- количество элементов в обоих списках должно быть одинаковым;
- должно существовать прямое соответствие между позицией одного и то же элемента в обоих списках. Поэтому I элемент списка значений должен относиться к I столбцу в списке столбцов, II – ко II столбцу и т.д.
- типы данных элементов в списке значений должны быть совместимы с типами данных соответствующих столбцов таблицы.

Пример 10. Добавить в таблицу ТОВАР новую запись.

INSERT INTO Товар (Название, Цена, Сорт, КодПроизводителя) VALUES(‘ Славянский’”, 12, высший, 1234)

Если столбцы таблицы ***ТОВАР*** указаны в полном составе и в том порядке, в котором они перечислены при создании таблицы ***ТОВАР***, оператор можно упростить.

INSERT INTO Товар VALUES (‘Славянский ‘, 12, высший, 1234)

ОПЕРАТОР УДАЛЕНИЯ

Оператор удаления

Формат оператора:

DELETE FROM <имя_таблицы>[WHERE <условие_отбора>]

Оператор удаления

Если предложение ***WHERE*** присутствует, удаляются записи из таблицы, удовлетворяющие условию отбора.

Если опустить предложение ***WHERE***, из таблицы будут *удалены все записи без предупреждения и без запроса на подтверждения*, однако сама таблица сохранится.

Оператор удаления

При удалении строк с помощью ***DELETE*** эти строки сохраняются в системных сегментах отката на случай восстановления. Это может потребовать значительного времени. Поэтому лучше использовать ***TRUNCATE*** для удаления всех данных.

Пример 11. Удалить все прошлогодние сделки

```
DELETE  FROM  Сделка
WHERE Year(Сделка.
Дата)=Year(GETDATE())-1
```

ОПЕРАТОР ОБНОВЛЕНИЯ

Оператор обновления

Формат оператора:

UPDATE имя_таблицы

SET имя_столбца = <выражение>[,...n]
[WHERE <условие_отбора>]

Оператор обновления

В предложении SET указываются имена одного и более столбцов, данные в которых необходимо изменить.

Выражение представляет собой новое значение соответствующего столбца и должно быть совместимо с ним по типу данных.

Пример 12. Увеличить цену товаров первого сорта на 25%.

```
UPDATE  Товар  SET Цена= Цена*1.25
WHERE  Сорт= ‘Первый’
```